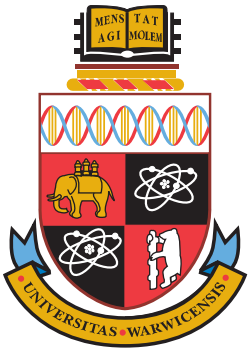


Extending the Bernoulli Factory to a Dice Enterprise

Giulio Morina

Thesis submitted for the degree of *Doctor of Philosophy*

University of Warwick
Department of Statistics

January 2021

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

Firstly I would like to thank Krzysztof Łatuszyński for the invaluable support and guidance given me during the course of my PhD. Every meeting with him was for me a reminder of his intelligence and brainpower, that never ceased to amaze me. I would also like to deeply thank Ewan Cameron for helping me especially during the initial phases of my PhD project.

I have received an unmatched support throughout the years from my family; they have always backed my decisions and I know they will in the future as well.

I am especially grateful for meeting my fellow OxWaSP companions, some of whom I ended up living with for the whole duration of my PhD. A shout out goes to my Botley crew companions: Virginia Aglietti, Jack Carter, Arne Gouwy, Petya Kindalova, Jeremias Knoblauch. I would like to especially thank Virginia and Jack, who somehow managed to put up with living with me for four years straight. A special thanks goes to my *siostra* Emilia Pompe, with whom I shared so many projects with. An honourable mention goes to Sam Davenport for his room-brightening laugh, which I thoroughly miss.

At Warwick I met so many wonderful people that made these 3 last years pass by in the blink of an eye. Among all people, I want to give my special thanks to Alejandra Avalos Pacheco, Elia Bisi, Panayiota Touloupou, Mattia Tosi, Nayia Constantinou, Matt Frost, Juan Ungredda, Jierong Luo. Finally, I'd like to thank all the people who (unintentionally) tried to boycott my PhD, but eventually failed: thanks to the in people in bridge/pub quiz club with whom I spent probably too much of my time at the department: Francesca Romana Crucinio, Jack Carter, Suzie Brown, Marco Palma, William Thomas, Ana Ignatieva. Finally, a special thanks goes to Pieralberto Guarniero (follow him at @pierpanda), who really tried his best to boycott my PhD during the pandemic - I am happy you did and our friendship grew stronger from it!

Declaration

I declare that this thesis is my own work, that I carried out under the supervision of Dr. Krzysztof Łatuszyński and Dr. Ewan Cameron, for the degree of Doctor of Philosophy in Statistics. I have not used sources or means without declaration in the text. I confirm that this thesis has not been submitted for a degree at any other university.

The contents of Chapter 4 and part of Chapter 1 are contained in the article **"From the Bernoulli Factory to a Dice Enterprise via Perfect Sampling of Markov Chains"** [41] which I have written during my PhD in collaboration with Dr. Krzysztof Łatuszyński, Dr. Piotr Nayar, Dr. Alex Wendland. This article has been accepted for publication on the Annals of Applied Probability and is yet to be published.

Abstract

The Bernoulli Factory problem consists of finding an algorithm that tosses an $f(p)$-coin, i.e. a coin that lands heads with probability $f(p)$ for a given function f, when p is unknown and the only source of randomness is given by a p-coin that can be tossed as many times as needed. This problem has been extensively studied and has found several applications in a variety of fields. In the present thesis we first propose novel methodologies for the construction of Bernoulli Factories algorithms that are applicable for a wide variety of functions $f(p)$. The main idea is to rephrase the original problem as constructing an unbiased estimator for $f(p)$ that is in $[0, 1]$ almost surely – a deed we propose to solve via debiasing techniques.

We also formally study a multivariate extension of the problem, named *Dice Enterprise*, that considers a more general setting where the sole source of randomness is given by an m-sided die and the aim is to roll v-sided dice, where the probability of rolling each face is a given function of the probabilities associated with the given m-sided die. We extend most of the current available theoretical results for Bernoulli Factories to this generic case. In particular, we characterise the class of functions for which a Dice Enterprise exists, provide an implementable algorithm based on polynomial approximations, and characterise functions that can admit a Dice Enterprise with a fast simulation. We also present a constructive way to build an efficient Dice Enterprise when the function of interest is a rational function with coefficients in $\mathbb{R}$. This construction uses techniques that have not been previously applied in this setting. We rephrase the original problem as simulating from the stationary distribution of a certain class of Markov chains - a task that we show can be achieved using perfect simulation techniques with the original m-sided die as the only source of randomness. We provide a comprehensive analysis of the running time of this new methodology.

1

Introduction

1.1 The Bernoulli Factory

The nomenclature *Bernoulli Factory* originated in the paper by Keane and O'Brien [30], itself motivated by a problem posed by Asmussen et al. [1] in the context of regenerative steady-state simulations [19]: given a stream of i.i.d. Bernoulli random variables of unknown parameter $p \in (0, 1/2)$, is it possible to simulate tosses of a coin that has probability $2p$ of landing heads? If such algorithm exists, we require for it to run in finite time a.s. for all possible admissible values of p. The problem can be stated more generally as follows, as proposed by Huber [23]:

Definition 1.1 (Bernoulli Factory)**.** Given an unknown $p \in S \subset [0, 1]$ and a known function $f(p) : S \subset [0, 1] \to [0, 1]$, let $\mathcal{A}$ be a computable function that takes as input a number $u \in [0, 1]$ and a sequence of values in $\{0, 1\}$, and returns an output in $\{0, 1\}$. For any sequence $\left(X^{(n)}\right)_{n\in\mathbb{N}^+} \overset{iid}{\sim} \text{Bern}(p)$ and $U \sim \text{Unif}(0, 1)$, let

$$T := \inf\left\{n : \mathcal{A}\left(U, X^{(1)}, X^{(2)}, \ldots\right) = \mathcal{A}\left(U, X^{(1)}, X^{(2)}, \ldots, X^{(n)}\right)\right\}.$$

Assume the following holds:

- T is a stopping time with respect to the natural filtration and is almost surely finite.
- $\mathcal{A}\left(U, X^{(1)}, X^{(2)}, \ldots\right) \sim \text{Bern}(f(p))$.

Then $\mathcal{A}$ is a *Bernoulli Factory* for f and T is its running time.

Therefore, a Bernoulli Factory is an algorithm that observes a random – albeit almost surely finite – number of tosses of a coin that has probability p of landing

heads and returns heads with probability $f(p)$. The algorithm is allowed to make use of an independent source of randomness, here represented by $U \sim \text{Unif}(0,1)$. This is to ease the notation and analysis, but it is not strictly necessary from a theoretical standpoint when $p \in (0,1)$ as some of the coin tosses can be used to generate U itself, as we will clarify later. If p is allowed to be exactly 0 or 1, an auxiliary source of randomness is needed.

The fundamental result by Keane and O'Brien [30], here reported, completely characterises the class of functions for which a Bernoulli Factory can be found.

Theorem 1.2 (Keane and O'Brien [30])**.** *Given $f : S \subset (0,1) \to [0,1]$, a Bernoulli Factory for f exists if and only if:*

- *$f(p)$ is continuous;*
- *Either f is constant or there exists $n_0 \in \mathbb{N}$ such that*

$$\min\left(f(p), 1 - f(p)\right) \geq \min(p, (1-p))^{n_0}, \tag{1.1}$$

for all $p \in S$.

The same statement was proved independently by Wästlund [61] using different techniques. Therefore, the theorem gives a negative answer to the question posed by Asmussen et al. [1]: consider $f(p) = 2p$ and set $p = 1/2 - \epsilon$, so that

$$\begin{aligned} \min\left(f(p), 1 - f(p)\right) &= 2\epsilon, \\ \min(p, (1-p))^{n_0} &= \left(\frac{1}{2} - \epsilon\right)^{n_0}. \end{aligned}$$

As we want eq. (1.1) to hold for all $p \in (0, 1/2)$, one can see that as $\epsilon \to 0$, the left hand side of the inequality tends to 0 while the right hand side is always greater than 0 for any choice of n_0.

Unfortunately, the proof of Theorem 1.2 does not provide an implementable procedure to design a Bernoulli Factory for any given f. This is why most of the subsequent research has been focusing on providing efficient Bernoulli Factory type of algorithms for either specific or generic functions f [18, 22, 23, 24, 32, 38, 39, 42, 43]. Arguably, the most famous example of a Bernoulli Factory – although rarely recognised as such – is that of Von Neumann [60] to construct a fair coin out of a biased one. The procedure goes as follows: toss the given coin twice; if it lands on the same face repeat the experiment, otherwise output the result of the first toss. One can see that this is a valid Bernoulli Factory for $f(p) = 1/2, p \in (0,1)$.

Bernoulli Factories have gained much attention in the recent years, as they have been successfully applied in a plethora of different fields. In statistics, they have been used to design MCMC algorithms that can tackle intractable likelihood models and perform Bayesian inference [16, 17, 20, 59], as well as to design particle filters in scenarios where weights are not available analytically [55]. In computer science they have found applications on the construction of Buffon machines, finite-state machines, and pushdown automata [14, 42]. In quantum physics, Bernoulli Factories are one of the few examples for which it is provable that a quantum computer has advantages over a traditional one [8, 47, 62]. In probability, they have been applied to perform exact simulations of diffusions [4, 32], develop perfect simulation algorithms [3, 15, 33], and to study the construction of unbiased estimators [27]. Other works that make use of Bernoulli Factories include reduction in mechanism design [6, 12, 44] and the multi-armed bandit problem [56].

1.2 Notation remarks

We now set up the notation that will be used throughout the thesis. We use bold symbols for vectors and, unless differently specified, we let vector indices start from 0. We define the open m dimensional probability simplex as

$$\Delta^m = \left\{ \boldsymbol{p} = (p_0, \dots, p_m) \in (0,1)^{m+1} : \sum_{i=0}^{m} p_i = 1 \right\},$$

and denote by $\bar{\Delta}^m$ its closure. It is also convenient to introduce the scaled by n discrete m dimensional simplex as

$$\Lambda_n^m = \left\{ \boldsymbol{k} = (k_0, \dots, k_m) \in \{0, 1, \dots, n\}^{m+1} : \sum_{i=0}^{m} k_i = n \right\}.$$

For $b \in \{0, \dots, m\}$, by $\boldsymbol{e_b} \in \bar{\Delta}^m$ denote the b^{th} standard unit vector, i.e. a vector of zeros with a 1 in the b^{th} position. Note that when we use the symbol $\subset$ the inclusion may not be strict; i.e. it holds that $A \subset A$.

We use a standard multi-index notation: given $\boldsymbol{p} \in \Delta^m$ and a vector $\boldsymbol{k}$ of length $(m+1)$ we write

$$\boldsymbol{p}^{\boldsymbol{k}} = p_0^{k_0} \cdot p_1^{k_1} \cdot \ldots \cdot p_m^{k_m},$$

and for a real number $k \in \mathbb{R}$ we denote $\boldsymbol{p}^k = p_0^k \cdot \ldots \cdot p_m^k$. Given $\boldsymbol{p} \in \mathbb{R}^{m+1}$ and $\boldsymbol{q} \in \mathbb{R}^{m+1}$ we write $\boldsymbol{p} \leq \boldsymbol{q}$ if and only if $p_i \leq q_i, \forall i \in \{0, \dots, m\}$. By $\|\cdot\|_1$ denote the

1-norm, so that $\|\boldsymbol{p}\|_1 = \sum_{i=0}^{m} p_i$. Finally, for $\boldsymbol{k} \in \Lambda_n^m$ we write

$$\binom{n}{\boldsymbol{k}} = \frac{n!}{k_0! \dots k_m!}.$$

We write $X \sim \boldsymbol{p}$ to indicate a sample from the categorical distribution of parameter $\boldsymbol{p} \in \Delta^m$ on $\Omega = \{0, \dots, m\}$, i.e. $\mathbb{P}_{\boldsymbol{p}}(X = i) = p_i$ for $i \in \{0, \dots, m\}$ and 0 otherwise. Equivalently, we refer to it as rolling a $\boldsymbol{p}$-die. Notice that we write $\mathbb{P}_{\boldsymbol{p}}$ with the subscript $\boldsymbol{p}$ to highlight that the probability function depends on a parameter $\boldsymbol{p}$, which will often be unknown. If the vector $\boldsymbol{p}$ is not known explicitly, but there is a mechanism to sample $X \sim \boldsymbol{p}$ (e.g. via experiment or computer code), we call this mechanism a black box to sample from $\boldsymbol{p} \in \Delta^m$. Alternatively, if we want to stress that the vector $\boldsymbol{\mu} \in \Delta^m$ is given explicitly, we refer to it as known distribution $\boldsymbol{\mu}$. For a vector valued random variable $\boldsymbol{X}$, we let $\mathbb{E}[\boldsymbol{X}] = (\mathbb{E}[X_0], \dots, \mathbb{E}[X_m])$; an estimator $\widehat{\boldsymbol{p}}$ of $\boldsymbol{p}$ will be unbiased if $\mathbb{E}[\widehat{\boldsymbol{p}}] = \boldsymbol{p}$.

We interchangeably use $(p_0, p_1) \in \Delta^1$ and $p \in (0, 1)$, identifying p as p_1. Analogously as the $\boldsymbol{p}$-die, we also use the terminology p-coin to indicate a mechanism that generates samples from a Bernoulli distribution of parameter p, identifying 1 as heads and 0 as tails.

1.3 Thesis outline

This thesis has three main objectives: (i) to provide a generic framework for the practical implementation of Bernoulli Factories; (ii) to extend the theory of the Bernoulli Factory to a multivariate setting named *Dice Enterprise*, i.e. considering Categorical distributions rather than Bernoulli; (iii) to study a novel and efficient algorithm to construct a Dice Enterprise for rational functions which makes use of perfect simulation techniques.

Chapter 2 is foremost an introduction to the main techniques that have been exploited in the literature to construct Bernoulli Factories. We first provide an overview of different linear Bernoulli Factories, i.e. algorithms targeting functions of the form $f(p) = Mp$, where $p \in (0, 1/M - \epsilon], \epsilon > 0$. We compare the most recently proposed algorithms, as linear Bernoulli Factories represent a fundamental building block for methodologies we will consider in the following. Importantly, this chapter also introduces novel techniques for designing Bernoulli Factories that arise by considering a different point of view: rather than aiming to find an algorithm that tosses an $f(p)$-coin, we can instead construct unbiased estimators for $f(p)$ satisfying specific properties. This change of perspective allows for novel approaches

to the construction of Bernoulli Factories that are based on debiasing techniques. Moreover, under this new setting, we can generalise the hypothesis of having access to a p-coin to just having a black-box outputting unbiased estimators of p, as this is commonly the case in statistical and probabilistic applications. We show how these novel techniques can be applied to design Bernoulli Factories when $f(p)$ admits specific series expansions, as well as showing how such methodologies can be applied more broadly on some other examples.

Chapter 3 introduces and formalises a multivariate extension of the Bernoulli Factory named *Dice Enterprise*. Rather than focusing on coins, that is Bernoulli distributions, we consider the more generic case of having access to dice, i.e. Categorical distributions. Multivariate extensions have been considered by several works [12, 16, 45, 55], mainly for specific algorithms, but the Dice Enterprise problem has not been systematically studied, especially from a theoretical perspective. We give a formal definition of a Dice Enterprise and extend most of the current available results on Bernoulli Factories to this case. In particular:

- We extend Theorem 1.2 to the multivariate setting, thus completely characterising the type of functions for which a Dice Enterprise exists;
- We generalise the approaches of Nacu and Peres [43] and Łatuszyński et al. [32], thus clarifying the connection that exists between a Dice Enterprise algorithm targeting a function f and the construction of polynomial envelopes for f. We also provide a practical and implementable Dice Enterprise when such envelopes are available;
- We define the class of functions for which a Dice Enterprise has a fast simulation, that is the probability on the number of rolls required by the algorithm to terminate decreases exponentially. This extends the work of Nacu and Peres [43]. We also provide novel efficient algorithms for functions that admit a series expansion;
- We highlight the connection that exists between a Dice Enterprise and the case where multiple independent coins are given, rather than a die, as this is common in applications [12, 16, 55]. We also give an initial look at extending this work further to dice with infinitely many faces, that is considering any discrete distribution.

Constructing a Dice Enterprise for a given function f is still generally challenging, and the strategies described in the previous chapters do not directly apply to all

possible function or can lead to sub-optimal implementations. Chapter 4 provides a constructive way to design a Dice Enterprise for rational functions f with coefficients in $\mathbb{R}$. Our construction can be applied to rational functions mapping between probability simplices

$$f : \Delta^m \to \Delta^v, \qquad m, v \geq 1, \tag{1.2}$$

Our approach relies on rephrasing the original problem as sampling from the stationary distribution of a suitably designed Markov chain. This is achieved by first decomposing the given rational function in a fashion inspired by Mossel et al. [42], and similarly based on Polya's theorem on homogeneous positive polynomials [50]. However, the decomposition is extended in such a way that as to allow to construct a Markov chain whose evolution can be simulated by just rolling the original die. We also allow for coefficients in $\mathbb{R}$ and derive our explicit construction for multivariate scenarios. Then, perfect simulations techniques, such as Coupling From The Past (CFTP) [52] or Fill's interruptible algorithm [13], can be employed to get a sample distributed precisely as its stationary distribution. Moreover, for $m = 1$ in eq. (1.2), that includes the classic Bernoulli Factory setting $m = v = 1$ as a special case, a monotonic version of CFTP is proposed, improving the efficiency of implementation. Under this scenario, we show that the method has a fast simulation (i.e. the required number of tosses has exponentially decaying tail probabilities) and the expected number of calls to the original die is linear in the degree of the resulting polynomial. To prove the result we demonstrate a fact of wider interest: the convolution of a Bin $\left(n, \frac{1}{2}\right)$ variable with any finite, integer valued random variable is log-concave when n is big enough.

In the remainder of this introductory section, we will examine how we can sample from any given known categorical distribution when the only source of randomness available is either a coin or a die with unknown bias. In general, one would get a sample from a categorical distribution by generating U from a Uniform distribution and producing an output by comparing U with the cumulative sums of the given pmf. In this case, we can use the given coin or die to generate bit by bit the digits of the uniform r.v., so to still get a sample from the desired categorical distribution in finite time.

1.4 Sampling from known categorical distributions

Sampling from a known distribution $\boldsymbol{\mu} = (\mu_0, \dots, \mu_k) \in \Delta^k$ is usually done by sampling $U \sim \text{Unif}(0, 1)$ and setting

$$Z = i \quad \text{if} \quad \sum_{j=0}^{i-1} \mu_j < U \leq \sum_{j=0}^{i} \mu_j.$$

Throughout the thesis, we will often assume that a generator of uniform random variables is available, as common in applications. However, this assumption is not restrictive from the theoretical viewpoint as we can use the given coin or die to generate uniform random variables to any arbitrary precision. In the spirit of [32, 43], we can consider B, the binary representation of U and notice that this is an i.i.d. sequence of Bern(1/2). Let $B_{1:l}$ denote the first l bits of U and let $(B_{1:l})_{10}$ be its representation in base 10. Clearly $(B_{1:l})_{10} \leq U \leq (B_{1:l})_{10} + 2^{-l}$, so that we could set

$$Z = i \qquad \text{if} \qquad \sum_{j=0}^{i-1} \mu_j \leq (B_{1:l})_{10} \quad \text{and} \quad (B_{1:l})_{10} + 2^{-l} \leq \sum_{j=0}^{i} \mu_j,$$

where l is big enough so that there exists an i such that the condition above is satisfied.

Therefore, we do not need to have access to a generator of uniform random variables to sample from a categorical distribution — it is enough to obtain a sequence of independent tosses of a fair coin. Algorithm 1 is a variation of Von Neumann's algorithm [60] that outputs a fair coin given access to an i.i.d. sequence of rolls of an arbitrary die.

Algorithm 1 Fair coins from a die

Input: black box to sample from $\boldsymbol{p} \in \Delta^m$.

Output: a sample from Bern(1/2).

1: Sample $X_1, X_2 \overset{iid}{\sim} \boldsymbol{p}$
2: **if** $X_1 < X_2$ **then** set $Y := 0$
3: **else if** $X_1 > X_2$ **then** set $Y := 1$
4: **else if** $X_1 = X_2$ **then** discard X_1, X_2 and GOTO 1
5: **end if**
6: **Output** Y

The algorithm can be further refined if a stream of fair coin tosses is required [48].

Consequently, given a black box to sample from a Categorical distribution of unknown parameter $\boldsymbol{p} \in \Delta^m$, Algorithm 2 outputs a sample from a known distribution $\boldsymbol{\mu} \in \Delta^k$.

Algorithm 2 Categorical distribution from a die

Input: black box to sample from $\boldsymbol{p} \in \Delta^m$.

Output: a sample from a known distribution $\boldsymbol{\mu} \in \Delta^k$.

1: Sample $Y \sim \text{Bern}(1/2)$ using Algorithm 1

2: If $l = 1$ set $B_{1:1} = Y$, otherwise set $B_{1:l} = B_{1:l-1}|Y$ (where | indicates bitwise concatenation)

3: **if** there exists an $i \in \{0, \dots, k\}$ such that

$$\sum_{j=0}^{i-1} \mu_j \leq (B_{1:l})_{10} < (B_{1:l})_{10} + 2^{-l} \leq \sum_{j=0}^{i} \mu_j$$

then set $Z := i$

4: **else**

5: Set $l = l + 1$ and GOTO 2

6: **end if**

7: **Output** Z

Remark 1.3. Sampling from a known categorical distribution $\boldsymbol{\mu}$ as in Algorithm 2 requires knowledge of $(\mu_0, \dots, \mu_k)$. It is however common, especially in the machine learning literature [28, 36], to have access to a parametrisation of $\boldsymbol{\mu}$ of the form $\boldsymbol{\gamma} = \log(\boldsymbol{\mu}) + \alpha$, where $\alpha \in \mathbb{R}$. One could theoretically recover the value of $\boldsymbol{\mu}$ by applying the inverse transformation and normalising accordingly. However, the Gumbel-max trick method [9, 10] can be applied to directly sample from $\boldsymbol{\mu}$. This approach consists of sampling $(k+1)$ independent uniform random variables U_i and outputting $\arg\max_{0 \leq i \leq k} \{\gamma_i - \log[-\log(U_i)]\}$. Note that we can apply Algorithm 1 to sample U_i to any arbitrary decision and therefore stop the algorithm as soon as a decision can be reached.

2

Designing Techniques for Bernoulli Factories

2.1 Introduction

The main aim of this chapter is to provide a generic framework for the design and practical implementation of Bernoulli Factory algorithms. In Section 2.2 we present the current state-of-the-art algorithms for linear Bernoulli Factories, which represent a fundamental building block for the methodologies we will develop. We compare different already available algorithms and discuss their advantages and drawbacks. We discuss and recap in Section 2.3 the common approaches taken to construct a Bernoulli Factory: via polynomial envelopes, via iterative algorithms, and via recursive algorithms. In Section 2.4 we propose novel approaches to construct Bernoulli Factories based on the key observation that tossing an $f(p)$-coin is analogous to obtaining unbiased estimators of $f(p)$ that are in $[0, 1]$ almost surely. We show in Sections 2.5 and 2.6 how to apply the proposed techniques to construct Bernoulli Factories for functions $f(p)$ that admit a positive or alternating power series representation. Finally, in Section 2.7 we use the proposed methodologies in the context of designing specific Bernoulli Factories. In Section 2.8 we make some final remarks and we present in Section 2.9 proofs of all the proposed results.

2.2 Linear Bernoulli Factories

A particularly interesting problem – mostly because of the challenges it poses and of its wide variety of applications – is constructing an efficient Bernoulli Factory for $f(p) = Mp$, where $M > 1$ is a fixed constant. As discussed in Section 1.1 and as a consequence of Theorem 1.2, this problem is unsolvable if p is allowed to be any number in $[0, 1/M)$. Perhaps surprisingly, the very same theorem guarantees that

if $p \in [0, 1/M - \epsilon], \epsilon > 0$, then a Bernoulli Factory does exist. In the following we will denote by N the number of tosses required for a Bernoulli Factory to output a result. The distribution of N can be used to analyse the algorithm efficiency.

Up to our knowledge, Nacu and Peres [43] (Theorem 1) were the first ones to derive an explicit algorithm for the case $M = 2$, although their construction can be extended to any arbitrary constant. Under the technical assumption $\epsilon \in (0, 1/8)$ (one can always set $\epsilon = 1/8$ if it is known that $2p < 1 - a$, with $a \geq 1/8$), their algorithm is fast, in the sense that there exist constants $C > 0$ and $\rho \in (0, 1)$ – which depend on ϵ but not on p – such that $\mathbb{P}_p(N > n) \leq C\rho^n$. Nevertheless, the proposed construction has two main drawbacks: (i) there is need to store sets of exponentially increasing size, and (ii) it requires an initial number of tosses of the p-coin which is often large, cf. Table 1. That means that if n_0 is the initial number of tosses required, then $\mathbb{P}(N < n_0) = 0$.

Problem (i) is solved when the proposed construction is paired up with the approach of Łatuszyński et al. [32]. An implementation of their algorithm in C++, which uses `Rcpp` to allow for an easy use in `R`, is made available at: `https://github.com/giuliomorina/LinearBF`. Flegal and Herbei [15] look into reducing the number of initial tosses required (cf. Table 1), although the proposed algorithm has worse performance and satisfies $\mathbb{E}[N] = \infty$. We shall also propose in Chapter 3 an algorithm that does not require an initial number of tosses, but still exhibits an infinite expected number of required tosses.

	Value of ϵ				
	0.49	0.4	0.25	0.1	0.01
Nacu and Peres [43]	N/A	N/A	N/A	2^{16}	2^{28}
Flegal and Herbei [15]	2^6	2^7	2^8	2^{11}	2^{18}
Example 3.12 in Chapter 3	0	0	0	0	0

Table 1: Comparison of initial number of tosses required for different doubling algorithms and for different values of ϵ.

It is only more recently that specialised algorithms have been derived, mainly by Huber (2016, 2017, 2019) [23, 24, 25], which do not require an initial number of tosses, are easily implementable, and have a fast implementation. Let N_{2016}, N_{2017} and N_{2019} be the number of tosses required for the corresponding proposed

algorithms to terminate. The following upper bounds are derived:

$$\begin{aligned}
&\mathbb{E}[N_{2016}] \leq 9.5M\epsilon^{-1}, \\
&\mathbb{E}[N_{2017}] \leq 7.57M\epsilon^{-1}, \\
&\mathbb{E}[N_{2019}] \leq 5.53M(\epsilon^{-1}+1).
\end{aligned} \tag{2.1}$$

It therefore appears that the latest algorithm [25] is generally the fastest, albeit being only an upper bound this is not necessarily the case. We compare empirically in Table 2, presented in the appendix, the performance of the three algorithms for different values of M and ϵ. Quite surprisingly the algorithm proposed by Huber [24], corresponding to N_{2017}, seems to perform worse than the other two, whilst the most recent algorithm performs generally better than the other ones. Only in the case small M and small ϵ, the algorithm proposed by Huber (2016) [23] seems to outperform the others. We also observe that the expected number of tosses appears to be linear in M and inversely proportional to ϵ, as expected. We notice that the empirical mean may not be the best criterion to compare the algorithms, as it is apparent that the distribution of the required number of p-coin tosses has heavy tails. Other estimators may be better suited for this case, see e.g. Lugosi and Mendelson [34].

2.3 Approaches to the construction of a Bernoulli Factory

We now introduce the most commonly used techniques to design a Bernoulli Factory: either via bounding the function f by appropriately chosen polynomials ('envelope method'), or by constructing an iterative or recursive algorithm.

Envelope method

This construction is based on a key observation by Nacu and Peres [43]: if a Bernoulli Factory for a function f exists, then such algorithm must define bounding polynomials for f. We will explore this connection further in Chapter 3.

Theorem 2.1 (Łatuszyński et al. [32], Nacu and Peres [43], Theorem 3.9)**.** *A Bernoulli Factory for $f(p) : S \subset (0,1) \to [0,1]$ exists if and only if for all $n \geq 1$*

there exist polynomials:

$$g^{(n)}(p) = \sum_{k=0}^{n} \binom{n}{k} a^{(n)}(k) p^k (1-p)^{n-k}, \qquad h^{(n)}(p) = \sum_{k=0}^{n} \binom{n}{k} b^{(n)}(k) p^k (1-p)^{n-k}$$

such that:

- $0 \leq a^{(n)}(k) \leq b^{(n)}(k) \leq 1$,
- $\lim_{n\to\infty} g^{(n)}(p) = f(p) = \lim_{n\to\infty} h^{(n)}(p)$,
- *For all $m < n$, it holds:*

$$a^{(n)}(k) \geq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} a^{(m)}(i), \qquad b^{(n)}(k) \leq \sum_{i=0}^{k} \frac{\binom{n-m}{k-i}\binom{m}{i}}{\binom{n}{k}} b^{(m)}(i).$$

The number of tosses N required for the algorithm to terminate satisfies $\mathbb{P}_p(N > n) = h^{(n)}(p) - g^{(n)}(p)$.

If such polynomials $g^{(n)}(p), h^{(n)}(p)$ are known, an implementable algorithm can be designed via the framework developed by Łatuszyński et al. [32] or as a special case of our construction in Section 3.3.1. Note that we can interpret $g^{(n)}(p)$ as the probability that at the n^{th} iteration the algorithm stops and returns 1, while $1 - h^{(n)}(p)$ is the probability that the algorithm stops and returns 0 at the n^{th} iteration.

Nacu and Peres [43] use this method to construct a Bernoulli Factory for $f(p) = \min(2p, 1-\epsilon)$. Quite surprisingly it turns out that such Bernoulli Factory is a fundamental building block, as it allows you to derive a Bernoulli Factory for all analytic functions. Other algorithms that make use of the same enveloping method are also provided by Nacu and Peres [43] for Lipschitz and twice differentiable functions. More recent works have looked into speeding up the proposed algorithm for $f(p) = \min(2p, 1-\epsilon)$ by providing tighter envelopes [15, 22, 58].

Iterative and recursive methods

A natural and perhaps more intuitive approach to design a Bernoulli Factory algorithm would be to have an algorithm that at the i^{th} iteration computes $Y_i \in \{-1, 0, 1\}$. If $Y_i = -1$, the algorithm proceeds to the next iteration, otherwise it

stops and outputs Y_i. Let Y be the final output of the algorithm, then:

$$\mathbb{P}_p(Y=1) = \sum_{k=0}^{\infty} \mathbb{P}_p\left(Y_k = 1\right) \prod_{i=0}^{k-1} \mathbb{P}_p\left(Y_i = -1\right).$$

Let I be the number of iterations required by the algorithm (possibly different than the number N of tosses needed), then:

$$\mathbb{P}_p(I > n) = \prod_{i=0}^{n} \mathbb{P}_p(Y_i = -1).$$

For the algorithm to be a valid Bernoulli Factory, it must be that $f(p) = \mathbb{P}_p(Y=1)$ and $\prod_{i=0}^{\infty} \mathbb{P}_p\left(Y_i = -1\right) = 0$ for all $p \in S$. This approach is arguably the most common one and has been taken to construct Bernoulli Factories for positive power series [38], alternating power series [32], and for specialised functions [12, 16, 55, 59].

The iterative approach is in principle equivalent to the envelope method. Indeed, if each iteration of the algorithm makes use of only one toss of the coin, the equivalence is straightforward:

$$g^{(n)}(p) = \sum_{k=0}^{n} \mathbb{P}_p(Y_k = 1) \prod_{i=0}^{k-1} \mathbb{P}_p(Y_i = -1),$$
$$h^{(n)}(p) = \sum_{k=0}^{n} \mathbb{P}_p(Y_k \neq 0) \prod_{i=0}^{k-1} \mathbb{P}_p(Y_i = -1).$$

However, in general, it is common that each iteration of the algorithm will make use of more coin tosses, so that the envelopes are then defined on an increasing sequence $\{n_i\}$. This still leads to a valid algorithm, as it can be shown that Theorem 2.1 still holds when the envelopes are defined on a subsequence.

Recursive algorithms are algorithms that are able to call themselves, possibly with a different set of parameters input. Although it is always possible to convert a recursive algorithm into an iterative one – so that the two approaches are in principle equivalent – the analysis may be easier and more intuitive when expressed in a recursive form. Indeed, Huber [25] proposes a "Fundamental Theorem of Perfect Simulation" that makes the analysis of recursive algorithms surprisingly simple. The same paper also makes use of this framework to construct and analyse Bernoulli Factories for linear functions.

Remark 2.2. Although all the above approaches are in principle equivalent, they do not exhaust the ways previous works have looked into constructing Bernoulli

Factories. For instance, in Chapter 4 we will derive a Bernoulli Factory algorithm via perfect simulation of Markov chains' stationary distributions, while Mossel et al. [42] make use of pushdown automata.

2.4 Bernoulli Factory as an unbiased estimator

We propose to introduce novel ways to construct a Bernoulli Factory by making the following key observation:

Lemma 2.3 (Łatuszyński et al. [32])**.** *Sampling random variables distributed as Bernoulli(p) is equivalent to constructing an unbiased estimator of p taking values in* $[0,1]$ *almost surely.*

Given an unbiased estimator $\widehat{p}$ of p that is in $[0,1]$ a.s., we can toss a p-coin by simulating $U \sim \text{Unif}(0,1)$ and output 1 if $U < \widehat{p}$ and 0 otherwise. This allows us to give an alternative, more general, definition of a Bernoulli Factory:

Definition 2.4 (Bernoulli Factory - Estimator version)**.** Given an unknown $p \in S \subset [0,1]$ and a known function $f(p) : S \subset [0,1] \to [0,1]$, let $\mathcal{A}$ be a computable function that takes as input a number $u \in [0,1]$ and a sequence of values in $[0,1]$, and returns an output in $[0,1]$. For any $U \sim \text{Unif}(0,1)$ and sequences $\left(X^{(n)}\right)_{n\in\mathbb{N}^+}$ of independent random variables such that $\mathbb{E}\left[X^{(n)}\right] = p$ and $\mathbb{P}_p\left(X^{(n)} \in [0,1]\right) = 1$, let

$$T := \inf\left\{n : \mathcal{A}\left(U, X^{(1)}, X^{(2)}, \ldots\right) = \mathcal{A}\left(U, X^{(1)}, X^{(2)}, \ldots, X^{(n)}\right)\right\}.$$

Assume the following holds:

- T is a stopping time with respect to the natural filtration and is almost surely finite.
- $\mathbb{E}\left[\mathcal{A}\left(U, X^{(1)}, X^{(2)}, \ldots\right)\right] = f(p)$;
- $\mathbb{P}_p\left(\mathcal{A}\left(U, X^{(1)}, X^{(2)}, \ldots\right) \in [0,1]\right) = 1$.

Then $\mathcal{A}$ is a *Bernoulli Factory* for f and T is its running time.

As discussed in Section 1.4, having access to a uniform random variable is not restrictive from a theoretical point of view, but it eases the analysis. Therefore, instead of trying to find an algorithm that directly tosses an $f(p)$-coin, we could aim to construct an unbiased estimate of $f(p)$ that takes value in $[0,1]$ for all possible $p \in S$. Moreover, instead of just assuming that we have access to a black-box mechanism

outputting 1 with probability p and 0 otherwise, we can just assume that we have a way to produce unbiased estimates of p that are in $[a, b]$ with $0 \leq a < b < \infty$ (we shall see later that the case $b > 1$ can still be of interest). This is commonly the case in statistical and probabilistic applications [16, 17, 32, 33]. Clearly, the classic Bernoulli Factory setting falls in this scenario, with $a = 0$ and $b = 1$.

This approach is strictly related to that of Jacob and Thiery [27], where the purpose is to provide sufficient and necessary conditions for the construction of non-negative unbiased estimators. In this case, we further require for such estimators to be bounded above by 1 and we look at this problem from an implementation perspective. In fact, this way to interpret a Bernoulli Factory suggests that we may try to use debiasing techniques to find a valid procedure. We will focus on the case where $f(p)$ can be written as a series:

$$f(p) = \sum_{k=0}^{\infty} \alpha_k(p), \tag{2.2}$$

and make use of two different debiasing techniques, a review of which can be found in Papaspiliopoulos [46], here referred to as importance sampling and Russian roulette.

2.4.1 Importance Sampling

To obtain an unbiased estimate of $f(p)$ as in eq. (2.2), we consider a proposal distribution $\boldsymbol{\beta}$ on $\mathbb{N}$ and proceed as follows:

1. Simulate $K \sim \boldsymbol{\beta}$,
2. Get an unbiased estimate $\widehat{\alpha}_K$ of $\alpha_K(p)$,
3. Output $\widehat{\alpha}_K / \beta_K$.

Consider the following assumptions:

(i.a) $\beta_k > 0$ if $\alpha_k(p) \neq 0$, $\forall k \in \mathbb{N}$ and $\sum_{k=0}^{\infty} \beta_k = 1$;

(ii) Being able to simulate K where $\mathbb{P}(K = k) = \beta_k$;

(iii) Being able to construct estimates $\widehat{\alpha}_k$ s.t. $\mathbb{E}[\widehat{\alpha}_k] = \alpha_k(p)$ for all $k \in \mathbb{N}$;

(iv.a) $\widehat{\alpha}_k \geq 0$, $\forall k \in \mathbb{N}$ a.s.;

(v.a) $\sup_k (\widehat{\alpha}_k / \beta_k) \leq 1$ a.s..

Proposition 2.5. *Assume (i.a), (ii), (iii). Then the importance sampling procedure produces an unbiased estimator of* $f(p) = \sum_{k=0}^{\infty} \alpha_k(p)$. *Assume further (iv.a), (v.a), then it is a valid Bernoulli Factory.*

The fact that assumptions (i.a), (ii), (iii) produce an unbiased estimator follows from the theory of importance sampling (see for instance Papaspiliopoulos [46], section 4.6.2). Additional assumptions (iv.a), (v.a) are required so for Lemma 2.3 to hold.

Remark 2.6. Usually the main challenge when implementing importance sampling is finding a proposal distribution that minimises the variance of the resulting estimator. The optimal choice would then be to set $\beta_k = \alpha_k(p)/f(p)$, although this is usually not possible as p is unknown. However, our main interest is not on the variance of the estimator, but rather to ensure that it is always bounded in $[0, 1]$. Clearly this has implications on the variance of the estimator, for instance Bhatia-Davis inequality [2] implies:

$$\mathrm{Var}\left[\frac{\widehat{\alpha}_K}{\beta_K}\right] \leq f(p)(1 - f(p)) \leq \frac{1}{4}.$$

It turns out, perhaps unsurprisingly, that importance sampling is closely related with the iterative method discussed in Section 2.3:

- *Converting an importance sampling algorithm to an iterative algorithm*

  Set:

  $$\mathbb{P}(Y_k = -1) = 1 - \frac{\beta_k}{1 - \sum_{j=0}^{k-1} \beta_j}, \qquad \mathbb{P}(Y_k = 1) = 1 - \frac{\alpha_k(p)}{1 - \sum_{j=0}^{k-1} \beta_j},$$

  and consequently $\mathbb{P}(Y_k = 0) = 1 - \mathbb{P}(Y_k = -1) - \mathbb{P}(Y_k = 1)$.

- *Converting an iterative sampling algorithm to one based on importance sampling*

  By reversing the relation above, we can set $\beta_k = (1 - \mathbb{P}(Y_k = -1)) \prod_{j=0}^{k-1} \mathbb{P}(Y_j = -1)$ and

  $$\widehat{\alpha}_k = \begin{cases} 1 & \text{with probability } \alpha_k(p) := \mathbb{P}(Y_k = 1) \prod_{j=0}^{k-1} \mathbb{P}(Y_j = -1), \\ 0 & \text{otherwise.} \end{cases}$$

  However, this implies that β_k may depend on p. Therefore, it is important to verify whether conditions (ii) and (v.a) still hold.

However, when converting an iterative algorithm to an importance sampling one, β_k may depend on p. Nevertheless, it is sometimes possible to consider a proposal distribution $\widetilde{\beta}_k \propto \beta_k$ which does not depend on p.

2.4.2 Russian Roulette

Another debiasing technique, which we will refer to as Russian roulette [35], has found applications in different statistical and mathematical settings [27, 37, 53]. Under our setting the procedure can be summarised as follows: consider again probabilities $\{\beta_k\}_{k\in\mathbb{N}}$ and denote $\omega_k = \sum_{i=k}^{\infty} \beta_i$. Then proceed as follows:

1. Simulate $K \sim \boldsymbol{\beta}$ (or analogously such that $\mathbb{P}(K \geq k) = \omega_k$);
2. Get unbiased estimates $\widehat{\alpha}_0, \ldots, \widehat{\alpha}_K$ of $\alpha_0(p), \ldots, \alpha_K(p)$;
3. Output $\sum_{i=0}^{K} \frac{\widehat{\alpha}_i}{\omega_i}$.

Consider the following further assumptions alongside conditions (ii)-(iii):

(i.b) $\beta_k \geq 0$, $\forall k \in \mathbb{N}$ and $\sum_{k=0}^{\infty} \beta_k = 1$,

(vi.a) $0 \leq \sum_{k=0}^{n} \frac{1}{\omega_k} \widehat{\alpha}_k \leq 1$, for all possible $\widehat{\alpha}_k$ and $n \in \mathbb{N}$, where $\omega_k = \sum_{i=k}^{\infty} \beta_i$.

Proposition 2.7. *Assume (i.b), (ii), (iii). Then the Russian roulette procedure produces an unbiased estimator of $f(p) = \sum_{k=0}^{\infty} \alpha_k(p)$. Assume further (vi.a), then the procedure is a valid Bernoulli Factory.*

A proof that Russian roulette procedure produces unbiased estimators can be found in Lyne et al. [35].

Remark 2.8. When designing a Bernoulli Factory for a function $f(p)$, we advice to first try using the importance sampling approach. This is usually easier to implement, as the conditions of Proposition 2.5 can usually be verified via simple calculations. Nevertheless, the Russian roulette approach can lead to different implementations and is generally more flexible, as it considers a sum of estimators, rather than a single term. This usually requires a more nuanced analysis to verify the algorithm correctness. In particular, when designing the algorithm, the terms $\alpha_k(p)$ may need to be carefully defined so to guarantee that all the conditions of Proposition 2.7 are valid.

2.4.3 Hypothesis relaxation

As will be clearer later, the hardest conditions to be satisfied are (v.a) and (vi.a), which ensure that the unbiased estimates obtained via importance sampling or Russian roulette are in $[0,1]$. It turns out that having extra knowledge on the function $f(p)$, namely $f(p) \leq 1-\epsilon$ for all $p \in S$ and a known $\epsilon > 0$, allows us to relax the two assumptions as:

(v.b) There exists $M \geq 0$ such that $\sup_k \frac{\widehat{\alpha_k}}{\beta_k} \leq M$.

(vi.b) There exists $M \geq 0$ such that $0 \leq \sum_{k=0}^{n} \frac{1}{\omega_k}\widehat{\alpha}_k \leq M$, for all possible $\widehat{\alpha}_k$ and $n \in \mathbb{N}$.

Indeed, if such M exists we can construct a Bernoulli Factory for $\widetilde{f}(p) := f(p)/M$ and then resort to a linear Bernoulli Factory of the form Mp to construct an $f(p)$-coin. We sum this up in the following Corollary:

Corollary 2.9. *Assume that $f(p) : S \subset [0,1] \to [0, 1-\epsilon]$ for a known ϵ and consider a sequence $\{\beta_i\}_{i\in\mathbb{N}}$.*

If conditions (i.a), (ii), (iii), (iv.a), (v.b), then the importance sampling technique can be used to construct a Bernoulli Factory for $f(p)$.

Otherwise, if conditions (i.b), (ii), (iii), (vi.b), then the Russian roulette procedure can be used to construct a Bernoulli Factory for $f(p)$.

Remark 2.10. Notice that the assumption $f(p) \leq 1-\epsilon$ for $\epsilon > 0$ cannot be dropped. Otherwise, one could toss the p-coin and output 0 if it returns tails and 2 otherwise. This satisfies conditions (i.a), (ii), (iii), (iv), (v.b) (with $\beta_0 = 1$, $\alpha_0(p) = 2p$, and $\alpha_k(p) = 0$ for $k > 0$) and so one could construct a Bernoulli Factory for $f(p) = 2p, p \in \left(0, \frac{1}{2}\right]$. Nevertheless it is known that this is impossible (cf. Section 1.1).

2.5 Bernoulli Factory for positive power series

We consider the case where $f(p) : S \subset [0,1] \to [0,1]$ can be represented as a positive power series centred at 0, that is:

$$f(p) = \sum_{k=0}^{\infty} a_k p^k, \quad a_k \geq 0, \forall k \geq 0.$$

This problem has already been considered by Mendo [38], under the additional assumption $\sum_{k=0}^{\infty} a_k = 1$. As we shall see, this condition can be easily relaxed and

the algorithm proposed by Mendo [38] can be derived (in a slightly more generalised version) as an iterative method.

In the following we will consider $M := \sum_{k=0}^{\infty} a_k b^k$, where recall that b is the maximum value that unbiased estimates of p can achieve, and it can be set equal to 1 in the classic Bernoulli Factory setting. Instead of deriving an algorithm targeting $f(p)$, we rather construct a Bernoulli Factory for $\widetilde{f}(p) := f(p)/M$ and then construct a Bernoulli Factory for $f(p)$ by:

- If $M \leq 1$, first toss an M-coin. If it results in heads toss and output a $\widetilde{f}(p)$-coin, otherwise outputs tails,
- If $M > 1$ and $f(p) \leq 1 - \epsilon$, we can resort to a linear Bernoulli Factory (cf. Corollary 2.9).

Let N be the number of tosses required for the algorithm to terminate and output a toss of an $\widetilde{f}(p)$-coin, we will use the proposed approaches to derive three algorithms:

- Iterative algorithm
  - *Assumptions:* classic Bernoulli Factory setting (i.e. $b \leq 1$), $M := \sum_{k=0}^{\infty} a_k < \infty$;
  - *Running time:* $\mathbb{E}[N] = \frac{M - f(p)}{M(1-p)}$;
  - *Note:* the frameworks produces an analogous algorithm to the one proposed by Mendo [38], although we no longer assume $M = 1$.
- Importance sampling
  - *Assumptions:* unbiased estimates of $p \in [0, b]$, $M := \sum_{k=0}^{\infty} a_k b^k < \infty$;
  - *Running time:* $\mathbb{E}[N] \leq \frac{1}{M} \sum_{k=1}^{\infty} k a_k b^k$;
  - *Note:* in the classic Bernoulli Factory case (i.e. $b \leq 1$), the running time can be made equal to the one of the iterative algorithm.
- Russian roulette
  - *Assumptions:* unbiased estimates of $p \in [0, b]$ where $b < 1$, $M := \sum_{k=0}^{\infty} a_k < \infty$;
  - *Running time:* $\mathbb{E}[N] \leq \frac{b}{1-b}$;
  - *Note:* due to the additional assumption, the algorithm cannot be used in the classic Bernoulli Factory setting (i.e. when only tosses of a p-coin are available).

2.5.1 Iterative algorithm

We consider the classic Bernoulli Factory setting, i.e. there exists a black-box mechanism that outputs 1 with unknown probability p and 0 otherwise. In this case we have to set $a = 0, b = 1$ and define $M := \sum_{k=0}^{\infty} a_k$; under the framework of Section 2.3, it must be:

$$\widetilde{f}(p) := \frac{1}{M} \sum_{k=0}^{\infty} a_k p^k = \mathbb{P}(Y = 1) = \sum_{k=0}^{\infty} \mathbb{P}\left(Y_k = 1\right) \prod_{i=0}^{k-1} \mathbb{P}\left(Y_i = -1\right).$$

This immediately suggests that we should set:

$$\mathbb{P}\left(Y_k = 1\right) \prod_{i=0}^{k-1} \mathbb{P}\left(Y_i = -1\right) = \frac{a_k}{M} p^k.$$

This can be achieved by setting:

$$\begin{aligned} \mathbb{P}\left(Y_0 = y_0\right) &= \begin{cases} 1 - \frac{a_0}{M} & \text{if } y_0 = -1 \\ 0 & \text{if } y_0 = 0 \\ \frac{a_0}{M} & \text{if } y_0 = 1 \\ 0 & \text{otherwise} \end{cases}, \\ \mathbb{P}\left(Y_i = y_i\right) &= \begin{cases} \frac{\sum_{j=i+1}^{\infty} a_j}{\sum_{j=i}^{\infty} a_j} p & \text{if } y_i = -1 \\ 1 - p & \text{if } y_i = 0 \\ \frac{a_i}{\sum_{j=i}^{\infty} a_j} p & \text{if } y_i = 1 \\ 0 & \text{otherwise} \end{cases}. \end{aligned} \tag{2.3}$$

We sum this up in the following proposition and in Algorithm 3.

Proposition 2.11. *Consider $f(p) : S \subset [0,1] \to [0,1]$ such that $f(p) := \sum_{k=0}^{\infty} a_k p^k$, $a_k \geq 0$ for all $k \geq 0$, and $M := \sum_{k=0}^{\infty} a_k < \infty$. Let $\widetilde{f}(p) := f(p)/M$ and denote by N the number of p-tosses needed to toss a $\widetilde{f}(p)$-coin. Then Algorithm 3 is a valid Bernoulli Factory for $\widetilde{f}(p)$ and:*

$$\mathbb{P}(N > n) = \frac{1}{M} \left(\sum_{k=n+1}^{\infty} a_k \right) p^n, \qquad \mathbb{E}[N] = \frac{M - f(p)}{M(1-p)}.$$

Algorithm 3 Bernoulli Factory for positive power series centred at 0 via iterative algorithm

$i \leftarrow 0$
$X \sim \text{Bern}(a_0/M)$
if $X = 1$ **then** $Y_0 \leftarrow 1$ **else** $Y_0 \leftarrow -1$
while $Y_i = -1$ **do**
    $i \leftarrow i + 1$
    $Y_i \sim \text{Bern}(p)$
    if $Y_i = 1$ **then**
        $X \sim \text{Bern}\left(\frac{a_i}{\sum_{j=i}^{\infty} a_j}\right)$
        if $X = 1$ **then** $Y_i = 1$ **else** $Y_i = -1$
    end if
end while
return Y_i

2.5.2 Importance sampling

The fact that $f(p)$ can be expressed as a positive power series, immediately suggests that we can consider probabilities $\beta_k \propto a_k b^k$ and with probability $K \sim \boldsymbol{\beta}$ output an unbiased estimate of $a_K p^K$. Algorithm 4 recaps the final algorithm and we prove its correctness in Proposition 2.12.

Algorithm 4 Bernoulli Factory for positive power series centred at 0 via importance sampling

Simulate K such that $\mathbb{P}(K = k) \propto a_k b^k$
Obtain K unbiased estimates $\widehat{p}_1, \ldots, \widehat{p}_K$ (or stop as soon one estimate is equal to 0)
return $\frac{\prod_{i=1}^{K} \widehat{p}_i}{b^K}$

Proposition 2.12. *Consider $f : S \subset [0,1] \to [0,1]$ such that $f(p) := \sum_{k=0}^{\infty} a_k p^k$, $a_k \geq 0$ for all $k \geq 0$, and $M := \sum_{k=0}^{\infty} a_k b^k < \infty$. Let $\widetilde{f}(p) := f(p)/M$ and denote by N the number of p-coin tosses needed to toss a $\widetilde{f}(p)$-coin. Then Algorithm 4 is a valid Bernoulli Factory for $\widetilde{f}(p)$ and:*

$$\mathbb{P}(N > n) \leq \frac{1}{M} \sum_{k=n+1}^{\infty} a_k b^k, \qquad \mathbb{E}[N] \leq \frac{1}{M} \sum_{k=1}^{\infty} k a_k b^k.$$

In order to improve the running time of the algorithm, note that we can stop generating unbiased estimators of p as soon as we observe a 0. In the classic Bernoulli

Factory case, which we can always impose if $b \leq 1$, this happens with probability $1 - p$. We show in the following corollary that the running time in this case is equivalent to that of Algorithm 3 obtained via the iterative procedure. This should not come as surprising, as a closer inspection reveals that the two algorithms are in fact equivalent: the random variable X in Algorithm 3 has the role of determining the value of K in Algorithm 4.

Corollary 2.13. *Consider $f(p)$ as in Proposition 2.12 and assume $M := \sum_{k=0}^{\infty} a_k < \infty$. Assume $b \leq 1$, so that it is possible to construct unbiased estimates of p that are in $\{0, 1\}$. Let N be the number of p-coin tosses required for the algorithm to terminate, then Algorithm 4 satisfies:*

$$\mathbb{P}(N > n) = \frac{p^n}{M} \sum_{k=n+1}^{\infty} a_k, \qquad \mathbb{E}[N] = \frac{M - f(p)}{M(1-p)}.$$

2.5.3 Russian roulette

Notice that the Russian roulette algorithm outputs $\sum_{i=0}^{K} \frac{\widehat{\alpha_i}}{\omega_i}$, and we know that $\widehat{\alpha_i} \leq a_i b^i$. Therefore, for condition (vi.b) to hold we need to define ω_i such that $\sum_{i=0}^{\infty} \frac{a_i b^i}{\omega_i}$ is bounded. This is usually not possible if $b > 1$. In the case where $b < 1$, we can consider $\omega_i = b^i$ under the additional assumption that $\sum_{k=0}^{\infty} a_k < \infty$. Other choices are possible, due to the flexibility of the debiasing technique, but they need to be tailored to the specific function f. We shall then explore this option and again consider the constant $M := \sum_{k=0}^{\infty} a_k$.

Algorithm 5 Bernoulli Factory for positive power series centred at 0 via Russian roulette

1: Simulate $K \sim \text{Geom}(1 - b)$ (starting from 0)
2: Obtain K unbiased estimates $\widehat{p}_1, \ldots, \widehat{p}_K$ (or stop as soon one estimate is equal to 0)
3: **return** $\sum_{i=0}^{K} \frac{1}{b^i} \frac{a_i}{M} \prod_{j=1}^{i} \widehat{p}^j$

Proposition 2.14. *Consider $f : S \subset [0, 1] \to [0, 1]$ such that $f(p) := \sum_{k=0}^{\infty} a_k p^k$, $a_k \geq 0$ for all $k \geq 0$, and $M := \sum_{k=0}^{\infty} a_k < \infty$. Assume it is possible to obtain unbiased estimates of p that are in $[0, b]$ a.s. with $b < 1$. Let $\widetilde{f}(p) := f(p)/M$ and denote by N the number of p-tosses needed to toss a $\widetilde{f}(p)$-coin. Then Algorithm 5 is a valid Bernoulli Factory for $\widetilde{f}(p)$ and:*

$$\mathbb{P}(N > n) \leq b^{n+1}, \qquad \mathbb{E}[N] \leq \frac{b}{1-b}.$$

2.5.4 Examples

Example 2.15. Consider:

$$f(p) = -\log(1-p) = \sum_{k=1}^{\infty} \frac{p^k}{k},$$

for $p \in S \subset [0,1]$ such that there exists ϵ satisfying $f(p) \leq 1-\epsilon, \forall p \in S$.

Notice that if $b \geq 1$, thus also in the classic Bernoulli Factory case, $M := \sum_{k=1}^{\infty} b^k/k = \infty$ and none of the proposed algorithms can be deployed. However, if $b < 1$, Algorithm 4 can be used, thus further motivating why extra information on p, in form of unbiased estimates that are not just coin tosses, may allow for an easier construction of a Bernoulli Factory. Therefore, assuming $b < 1$, the expected number N of unbiased estimates required to construct a Bernoulli Factory for $f(p)$, paired up with the bound of eq. (2.1) for linear Bernoulli Factories, is then:

$$\mathbb{E}[N] \leq -\frac{5.53Mb}{\log(1-b)(1-b)} \left(\epsilon^{-1} + 1\right).$$

Example 2.16. Given a p-coin with $p \in [0,1]$, we aim to toss a $f(p)$-coin where f is given by:

$$f(p) = 1 - \sqrt{1-p} = \sum_{k=1}^{\infty} \binom{2k-2}{k-1} \frac{1}{k2^{2k-1}} p^k$$

If $b < 1$, we can resort to both Algorithm 4, with $M := 1 - \sqrt{1-b}$, and Algorithm 5, with $M := 1$. The running times to produce tosses of an $f(p)$-coin of the two algorithms are given by:

$$\mathbb{E}[N_{IS}] = \sum_{k=1}^{\infty} \binom{2k-2}{k-1} \frac{1}{2^{2k-1}} b^k = \frac{b}{2\sqrt{1-b}}, \qquad \mathbb{E}[N_{RR}] = \frac{b}{1-b},$$

for the importance sampling and the Russian roulette schemes respectively.

If $b = 1$, we can no longer use the proposed Russian roulette approach, but we can resort to the iterative scheme, which is equivalent to the importance sampling one in terms of expected number of tosses required. We now have $M := 1$ and the running time is given by:

$$\mathbb{E}[N_{IT}] = \mathbb{E}[N_{IS}] = \frac{\sqrt{1-p}}{1-p}.$$

If $b > 1$, then $M = \infty$ and none of the proposed scheme can be used.

Example 2.17. Consider $f(p) : S \subset [0,1] \to [0,1]$ given by:

$$f(p) = e^p - 1 = \sum_{k=1}^{\infty} \frac{p^k}{k!},$$

and $p \in S$ such that $f(p) \leq 1 - \epsilon$. Therefore, $M := \sum_{k=1}^{\infty} \frac{b^k}{k!} = e^b - 1$ for the importance sampling scheme and $M := \sum_{k=1}^{\infty} \frac{1}{k!} = e - 1$ for the iterative and Russian roulette ones. The running time required by the three proposed methods to generate tosses of a $\widetilde{f}(p) := f(p)/M$-coin is then given by:

$$\begin{aligned}
\mathbb{E}[N_{IA}] &= \mathbb{E}[N_{IS}] = \frac{e^b - e^p}{(e^b - 1)(1-p)} && \text{if } b = 1 \\
\mathbb{E}[N_{IS}] &= be^b && \text{if } b > 0 \\
\mathbb{E}[N_{RR}] &= \frac{b}{1-b} && \text{if } b \in [0,1)
\end{aligned}$$

2.6 Bernoulli Factory for alternating power series

We now consider the case where $f(p) : S \subset [0,1] \to [0,1]$ can be written as an alternating power series centred at 0:

$$f(p) = \sum_{k=0}^{\infty} (-1)^k a_k p^k, \qquad a_k \geq 0, \forall k \geq 0.$$

As before, we shall consider a constant M and construct a Bernoulli Factory for $\widetilde{f}(p) := f(p)/M$. We can then recover tosses of an $f(p)$-coin in the same fashion as in Section 2.5.

Let N be the number of tosses required for the algorithm to terminate and output a toss of an $\widetilde{f}(p)$-coin. The proposed approaches will give rise to four different algorithms:

- Envelope method ($M = a_0$)
  - *Assumptions:* classic Bernoulli Factory setting (i.e. $b = 1$), decreasing coefficients $a_0 \geq a_1 \geq \ldots$;
  - *Running time:* $\mathbb{E}[N] = \frac{1}{a_0} \sum_{k=0}^{\infty} a_k p^k$;
  - *Note:* the algorithm has been proposed by Łatuszyński et al. [32].

- Iterative algorithm ($M = a_0$)
  - *Assumptions:* classic Bernoulli Factory setting (i.e. $b = 1$), decreasing coefficients $a_0 \geq a_1 \geq \ldots$,
  - *Running time:* $\mathbb{E}[N] = 1 + \frac{2}{a_0} \sum_{k=1}^{\infty} a_{2k-1} p^{2k-1}$,
  - *Note:* the hypothesis of decreasing coefficients can be relaxed if $p \in S$ is such that $\frac{a_{i+1}}{a_i} p \leq 1 - \epsilon_i$ for known $\epsilon_i > 0$.
- Importance sampling ($M = \sum_{k=0}^{\infty} a_{2k} b^{2k} - \sum_{k=0}^{\infty} a_{2k+1} a^{2k+1}$)
  - *Assumptions:* M finite, $b \leq \max_k \frac{a_{2k}}{a_{2k+1}}$, and $\frac{a_{2k}}{a_{2k+1}} \geq \frac{a^{2k+1}}{b^{2k}}$;
  - *Running time:* $\mathbb{E}[N] \leq \frac{1}{M} \sum_{n=0}^{\infty} \sum_{k=\lceil n/2 \rceil}^{\infty} (a_{2k} b^{2k} - a_{2k+1} a^{2k+1})$;
  - *Note:* In the classic Bernoulli Factory setting the assumption can be rephrased as requiring $M := \sum_{k=0}^{\infty} a_{2k} < \infty$ and the coefficients to be decreasing, i.e. $a_0 \geq a_1 \geq \ldots$. The running time is then equal to $\mathbb{E}[N] = \frac{1}{M(1-p)} \sum_{k=0}^{\infty} a_{2k}(1 - p^{2k+1})$.
- Russian roulette ($M = a_0$)
  - *Assumptions:* classic Bernoulli Factory setting, decreasing coefficients tending to 0, i.e. $a_0 \geq a_1 \geq \ldots$, $a_i \to 0$;
  - *Running time:* $\mathbb{E}[N] = \frac{1}{a_0} \sum_{k=0}^{\infty} a_{k+1} p^k$.

At first glance we notice that the algorithms arising from the envelope method and the iterative procedure have the same hypothesis, but different running times. The iterative algorithm appears to be more flexible and – under additional assumptions – the hypothesis of decreasing coefficients can be dropped. The algorithm produced via the Russian roulette, under the additional hypothesis for the coefficients to tend to 0, has a running time that is a small improvement over the one of the envelope method.

2.6.1 Envelope method

An algorithm that makes use of the enveloping polynomials (cf. Section 2.3) has been proposed by Łatuszyński et al. [32] (Proposition 3.4). The original formulation assumes $1 \geq a_0 \geq a_1 \geq \ldots$; we notice that we can consider $\widetilde{f}(p) = f(p)/a_0$ and just assume that the power series representation admits decreasing coefficients. For an

easier analysis, their proposed algorithm can be formulated as an iterative algorithm under the framework discussed in Section 2.3 by setting:

$$\mathbb{P}(Y_0 = y_0) = \begin{cases} 1 & \text{if } y_0 = -1 \\ 0 & \text{otherwise} \end{cases},$$

$$\mathbb{P}(Y_k = y_k) = \begin{cases} \frac{a_k}{a_{k-1}} p & \text{if } y_k = -1 \\ 1 - \frac{a_k}{a_{k-1}} p & \text{if } y_k = 0 \text{ and } k \text{ is even} \\ 1 - \frac{a_k}{a_{k-1}} p & \text{if } y_k = 1 \text{ and } k \text{ is odd} \\ 0 & \text{otherwise} \end{cases},$$

One can easily check that this is a valid choice:

$$\begin{aligned} \sum_{k=0}^{\infty} \mathbb{P}(Y_k = 1) \prod_{i=0}^{k-1} \mathbb{P}(Y_i = -1) &= \sum_{k=0}^{\infty} \left(1 - \frac{a_{2k+1}}{a_{2k}} p\right) \frac{a_{2k}}{a_0} p^{2k} \\ &= \frac{1}{a_0} \sum_{k=0}^{\infty} a_{2k} p^{2k} - a_{2k+1} p^{2k+1} = \frac{f(p)}{a_0} = \widetilde{f}(p). \end{aligned}$$

The number of tosses N required for the algorithm to terminate satisfies:

$$\mathbb{P}(N > n) = \frac{a_n}{a_0} p^n, \qquad \mathbb{E}[N] = \frac{1}{a_0} \sum_{k=0}^{\infty} a_k p^k.$$

2.6.2 Iterative algorithm

We propose an alternative iterative procedure. Proceeding in a similar fashion as in Section 2.5.1 and under the framework of Section 2.3, it must be:

$$\widetilde{f}(p) := \frac{1}{M} \sum_{k=0}^{\infty} (-1)^k a_k p^k = \mathbb{P}(Y = 1) = \sum_{k=0}^{\infty} \mathbb{P}(Y_k = 1) \prod_{i=0}^{k-1} \mathbb{P}(Y_i = -1).$$

We can therefore aim to set:

$$\widetilde{f}(p) = \underbrace{\frac{a_0}{M} - \frac{a_1}{M} p}_{\mathbb{P}(Y_0=1)} + \underbrace{\frac{a_2}{M} p^2 - \frac{a_3}{M} p^3}_{\mathbb{P}(Y_1=1)\mathbb{P}(Y_0=-1)} + \underbrace{\frac{a_4}{M} p^4 - \frac{a_5}{M} p^5}_{\mathbb{P}(Y_2=1)\mathbb{P}(Y_0=-1)\mathbb{P}(Y_1=-1)} + \dots$$

Clearly, we need to tune M and make additional assumptions in order to have valid probabilities. In particular, we can set $M = a_0$ and thus have:

$$\begin{aligned}
\mathbb{P}(Y_0 = y_0) &= \begin{cases} \frac{a_1}{a_0}p & \text{if } y_0 = -1 \\ 0 & \text{if } y_0 = 0 \\ 1 - \frac{a_1}{a_0}p & \text{if } y_0 = 1 \\ 0 & \text{otherwise} \end{cases}, \\
\mathbb{P}(Y_k = y_k) &= \begin{cases} \frac{a_{2k+1}}{a_{2k-1}}p^2 & \text{if } y_k = -1 \\ 1 - \frac{a_{2k}}{a_{2k-1}}p & \text{if } y_k = 0 \\ \frac{p}{a_{2k-1}}\left(a_{2k} - a_{2k+1}p\right) & \text{if } y_k = 1 \\ 0 & \text{otherwise} \end{cases},
\end{aligned} \tag{2.4}$$

where we need to make additional assumption for eq. (2.4) to be a valid pmf, as detailed in Proposition 2.18. Algorithm 6 summarises the procedure.

Algorithm 6 Bernoulli Factory for alternating power series centred at 0

Set $k := 1$ and toss $X \sim \text{Bern}\left(\frac{a_1}{a_0}p\right)$
if $X = 1$ **then** set $Y = -1$ **else** set $Y = 1$
while $Y = -1$ **do**
    Toss $X \sim \text{Bern}\left(\frac{a_{2k}}{a_{2k-1}}p\right)$ and $W \sim \text{Bern}\left(\frac{a_{2k+1}}{a_{2k}}p\right)$
    if $X = 0$ **then**
        Set $Y = 0$
    else if $W = 0$ **then**
        Set $Y = 1$
    end if
    Let $k = k + 1$
end while
return Y

Proposition 2.18. *Consider $f : S \subset [0,1] \to [0,1]$ such that $f(p) := \sum_{k=0}^{\infty}(-1)a_k p^k$, $a_k \geq 0$ for all $k \geq 0$, $a_0 \geq a_1 \geq \ldots$. Let $\widetilde{f}(p) := f(p)/a_0$ and denote by N the number of p-tosses needed to toss a $\widetilde{f}(p)$-coin. Then Algorithm 6 is a valid Bernoulli Factory for $\widetilde{f}(p)$ and:*

$$\mathbb{P}(N > n) = \frac{1}{a_0}\sum_{k=\lceil n/2 \rceil}^{\infty} p^{2k-1}(a_{2k-1} - a_{2k+1}p^2), \qquad \mathbb{E}[N] = 1 + \frac{2}{a_0}\sum_{k=1}^{\infty} a_{2k-1}p^{2k-1}.$$

Remark 2.19. Requiring decreasing coefficients is necessary to ensure that $\frac{a_{i+1}}{a_i}p \in [0,1]$ for all $p \in [0,1]$ and that a Bernoulli with such probability can be easily simulated. However, the assumption can be relaxed to just assume $\frac{a_{i+1}}{a_i}p \leq 1-\epsilon_i$, for a known $\epsilon_i > 0$ and for all $p \in S$. If this is the case, a $\frac{a_{i+1}}{a_i}p$-coin can be tossed by resorting to a linear Bernoulli Factory (cf. Section 2.2). Note that this will clearly change the analysis on the distribution of N, i.e. the number of tosses required by the algorithm.

2.6.3 Importance sampling

Recall that we assume having access to a procedure producing unbiased estimates of p that are in $[a,b]$ almost surely, with $a \geq 0$. We can aim to split the series into chunks that are non-negative almost surely, to this end we shall assume:

$$a_0 - a_1 p \geq 0, a_2 p^2 - a_3 p^3 \geq 0, \dots \qquad \forall p \in [a,b], \tag{2.5}$$

a necessary condition being $f(p) \geq 0, \forall p \in [a,b]$. The main idea is then similar to the one applied to positive power series: consider probabilities $\{\beta_i\}_{i\in\mathbb{N}}$ and output an unbiased estimate of $a_{2K}p^{2K} - a_{2K+1}p^{2K+1}$ with probability $K \sim \boldsymbol{\beta}$. The probabilities $\boldsymbol{\beta}$ need to be selected so that all the required conditions of Section 2.4.1 are met, giving rise to Algorithm 7. Its correctness is proved in Proposition 2.20, while in Corollary 2.22 we explore in greater details the classic Bernoulli Factory case.

Algorithm 7 Bernoulli Factory for alternating power series centred at 0 via importance sampling

1: Simulate K such that $\mathbb{P}(K=k) \propto \left(a_{2k}b^{2k} - a_{2k+1}a^{2k+1}\right)$
2: Obtain $2K+1$ unbiased estimates $\widehat{p}_1, \dots, \widehat{p}_{2K+1}$ (or stop as soon as one estimate is equal to 0)
3: **return** $\frac{a_{2k}\prod_{i=1}^{2k}\widehat{p}_i - a_{2k+1}\prod_{i=1}^{2k+1}\widehat{p}_i}{a_{2k}b^{2k} - a_{2k+1}a^{2k+1}}$

Proposition 2.20. *Consider $f : S \subset [0,1] \to [0,1]$ such that $f(p) := \sum_{k=0}^{\infty}(-1)^k a_k p^k$. Assume that $M := \sum_{k=0}^{\infty} a_{2k}b^{2k} - \sum_{k=0}^{\infty} a_{2k+1}a^{2k+1}$ is finite, $b \leq \max_k \frac{a_{2k}}{a_{2k+1}}$, and $\frac{a_{2k}}{a_{2k+1}} \geq \frac{a^{2k+1}}{b^{2k}}$. Let $\widetilde{f}(p) := f(p)/M$ and denote by N the number of p-tosses needed to toss a $\widetilde{f}(p)$-coin. Then Algorithm 7 is a valid Bernoulli Factory for $\widetilde{f}(p)$ and:*

$$\mathbb{P}(N > n) \leq \frac{1}{M}\sum_{k=\lceil n/2\rceil}^{\infty}(a_{2k}b^{2k} - a_{2k+1}a^{2k+1}).$$

Remark 2.21. Notice that the assumption $\frac{a_{2k}}{a_{2k+1}} \geq \frac{a^{2k+1}}{b^{2k}}$ can always be satisfied, as we can make a arbitrary close or equal to 0. However, this will have an impact on the value of M.

Corollary 2.22. *Consider $f(p)$ as in Proposition 2.20 and assume $b \leq 1$, so that it is possible to construct unbiased estimates of p that are in $\{0,1\}$. Assume that the power series coefficients are decreasing, i.e. $a_0 \geq a_1 \geq a_2 \geq \ldots$. Let $M := \sum_{k=0}^{\infty} a_{2k}$ and, assuming M is finite, let N be the number of p-coin tosses required for Algorithm 7 to terminate, then:*

$$\mathbb{P}(N > n) = \frac{p^n}{M} \sum_{k=\lceil n/2 \rceil}^{\infty} a_{2k}, \qquad \mathbb{E}[N] = \frac{1}{M(1-p)} \sum_{k=0}^{\infty} a_{2k}(1 - p^{2k+1}).$$

Remark 2.23. More in general, we can relax the assumption of eq. (2.5) and require the existence of an infinite increasing sequence $\{m_1, m_2, \ldots\}$ of non-negative integers such that:

$$\sum_{k=0}^{m_1} (-1)^k a_k p^k \geq 0, \sum_{k=m_1+1}^{m_2} (-1)^k a_k p^k \geq 0, \ldots \qquad \forall p \in [a,b].$$

This will require a small modification of Algorithm 7.

2.6.4 Russian roulette

An algorithm based on the Russian roulette technique arises quite naturally, as such technique exploits a random truncation argument of the series. The final algorithm is presented in the following.

Algorithm 8 Bernoulli Factory for alternating power series centred at 0 via Russian roulette

1: Simulate K such that $\mathbb{P}(K \geq k) = \frac{a_k}{a_0}$
2: Obtain K unbiased estimates $\widehat{p}_1, \ldots, \widehat{p}_K$ and convert them into tosses. Stop as soon one toss is equal to 0.
3: Let $\widehat{\alpha}_0 = 1$ and for $i = 1, \ldots, K$ let $\widehat{\alpha}_i = (-1)^i \frac{a_i}{a_0} \prod_{j=1}^{i} \widehat{p}_j$
4: Output $\sum_{i=0}^{K} \frac{a_0}{a_i} \widehat{\alpha}_i$

Proposition 2.24. *Consider $f : S \subset [0,1] \to [0,1]$ such that $f(p) := \sum_{k=0}^{\infty} (-1) a_k p^k$, $a_k \geq 0$ for all $k \geq 0$, $a_0 \geq a_1 \geq \ldots$, and $a_k \to 0$ as $k \to \infty$. Moreover, assume $b \leq 1$, so that it is possible to construct unbiased estimates of p that are in $\{0,1\}$ Let*

$\widetilde{f}(p) := f(p)/a_0$ and denote by N the number of p-tosses needed to toss a $\widetilde{f}(p)$-coin. Then Algorithm 8 is a valid Bernoulli Factory for $\widetilde{f}(p)$ and:

$$\mathbb{P}(N > n) = \frac{a_{n+1}}{a_0} p^n, \qquad \mathbb{E}[N] = \frac{1}{a_0} \sum_{n=0}^{\infty} a_{n+1} p^n.$$

2.6.5 Examples

Example 2.25. Consider

$$f(p) = e^{-p} = \sum_{k=0}^{\infty} (-1)^k \frac{p^k}{k!},$$

for $p \in [0, 1]$. Let N_{EM}, N_{IT}, N_{IS}, and N_{RR} be the number of tosses required by the different algorithms for the envelope method, iterative algorithm, importance sampling and Russian roulette respectively.

In the classic Bernoulli Factory setting, the envelope method, iterative algorithm and Russian roulette target $\widetilde{f}(p) = f(p)$ and have expected running times equal to:

$$\begin{aligned}
\mathbb{E}[N_{EM}] &= \sum_{k=0}^{\infty} \frac{p^k}{k!} = e^p, \\
\mathbb{E}[N_{IT}] &= 1 + 2 \sum_{k=1}^{\infty} \frac{p^{2k-1}}{(2k-1)!} = 2 \sinh(p) + 1, \\
\mathbb{E}[N_{RR}] &= \sum_{k=0}^{\infty} \frac{p^k}{(k+1)!} = \frac{e^p - 1}{p},
\end{aligned}$$

and therefore

$$\mathbb{E}[N_{RR}] \leq \mathbb{E}[N_{EM}] \leq \mathbb{E}[N_{IT}].$$

The importance sampling technique considers $M := \sum_{k=0}^{\infty} \frac{b^{2k}}{(2k)!} - \sum_{k=0}^{\infty} \frac{a^{2k+1}}{(2k+1)!} = \cosh(b) - \sinh(a)$ and targets $\widetilde{f}(p) = f(p)/M$. In the case where $M > 1$ (as for instance in the Bernoulli Factory case, where $a = 0$, $b = 1$) it is then not convenient to use this scheme. If a, b are such that $M \leq 1$, then this scheme may take fewer iterations.

Example 2.26. Consider

$$f(p) = 1 - \log(1 + p) = 1 - \sum_{k=1}^{\infty} \frac{p^k}{k},$$

for $p \in [0,1]$ and let N_{EM}, N_{IT}, N_{IS}, and N_{RR} as above.

The envelope method, iterative algorithm and Russian roulette target $\widetilde{f}(p) = f(p)$ and have expected running times equal to:

$$\begin{aligned}
\mathbb{E}[N_{EM}] &= 1 + \sum_{k=1}^{\infty} \frac{p^k}{k} = 1 - \log(1-p), \\
\mathbb{E}[N_{IT}] &= 1 + 2\sum_{k=1}^{\infty} \frac{p^{2k-1}}{2k-1} = 1 + 2\tanh^{-1}(p), \\
\mathbb{E}[N_{RR}] &= 1 + \sum_{k=0}^{\infty} \frac{p^k}{k+1} = -\frac{\log(1-p)}{p},
\end{aligned}$$

and note that the running time goes to infinity as $p \to 1$.

The importance sampling technique considers $M := \sum_{k=0}^{\infty} \frac{b^{2k}}{2k} - \sum_{k=0}^{\infty} \frac{a^{2k+1}}{2k+1}$, which is finite only if $b < 1$ and equal to $a - \tanh^{-1}(a) - \frac{1}{2}\log(1-b^2) - a + 1$. Therefore, unless a and b are such that $M \leq 1$, using this scheme is not convenient.

2.7 Specialised Bernoulli Factories

We now look at how the proposed techniques can be applied to specific examples, some of which have been considered in other works.

2.7.1 Bernoulli Factory for division

Given a p_0-coin and a p_1-coin, assume $p_1 - p_0 \geq \epsilon$ for a known $\epsilon > 0$. We wish to design an algorithm that tosses a (p_0/p_1)-coin. This problem has been considered by Nacu and Peres [43], but their proposed construction is quite involved. On the other hand, we now show how we can use the iterative scheme to derive a simple algorithm. We can write:

$$\frac{p_0}{p_1} = \frac{1}{2}p_0 \sum_{k=0}^{\infty} \left(1 - \frac{1}{2}p_1\right)^k,$$

so that under the framework of Section 2.3 we can set:

$$\mathbb{P}(Y_k = y_k) = \begin{cases} 1 - \frac{1}{2}p_1 & \text{if } y_k = -1 \\ \frac{1}{2}(p_1 - p_0) & \text{if } y_k = 0 \\ \frac{1}{2}p_0 & \text{if } y_k = 1, \end{cases}$$

and therefore have

$$\sum_{k=0}^{\infty} \mathbb{P}(Y_k = 1) \prod_{i=0}^{k-1} \mathbb{P}(Y_i = -1) = \frac{1}{2} p_0 \sum_{k=0}^{\infty} \left(1 - \frac{1}{2} p_1\right)^k = \frac{p_0}{p_1}.$$

Algorithm 9 recaps the procedure. Note that to toss a $(p_1 - p_0)$-coin we need to resort to a linear Bernoulli Factory; the detailed procedure is exposed in the proof of the following proposition. The final algorithm can also be seen a special of the 2-coin algorithm proposed by Gonçalves et al. [16].

Proposition 2.27. *Given a p_0-coin and a p_1-coin, assume $p_1 - p_0 \geq \epsilon$ and let N be the number of tosses required. Then, there exists a Bernoulli Factory for $(p_1 - p_0)$ which satisfies $\mathbb{E}[N] \leq 11.06(1 + \epsilon^{-1})$.*

Proof. The construction of the Dice Enterprise follows the idea of the proof of Proposition 14 of Nacu and Peres [43]. We first construct a Bernoulli Factory for $h(p_0, p_1) = \frac{1-p_1+p_0}{2}$: with probability equal to $1/2$ we return the toss of a p_0-coin, otherwise we toss a p_1-coin and reverse the flip, i.e. return heads if it lands tails and vice-versa. We then apply a linear Bernoulli Factory with $M = 2$, noticing: $2h(p_0, p_1) = 1 - p_1 + p_0 \leq 1 - \epsilon$. Finally, we flip the result. □

Algorithm 9 Bernoulli Factory for division

1: Let $X \sim \text{Bern}(1/2)$
2: **if** $X = 0$ **then**
3: Let $W \sim \text{Bern}(p_0)$
4: **if** $W = 1$ **then** stop and output 1 **else** GOTO 1
5: **else**
6: Let $W \sim \text{Bern}(p_1 - p_0)$ (cf. Proposition 2.27)
7: **if** $W = 1$ **then** stop and output 0 **else** GOTO 1
8: **end if**

2.7.2 Rational functions

Consider a rational function given by $f(p)/g(p)$, where both $f(p)$ and $g(p)$ are polynomial. In Chapter 4, we will study this problem in greater details, and even extend it to the case where f and g are functions of dice probabilities. Nevertheless, we can derive an alternative algorithm that makes use of the division algorithm we have just proposed. We manipulate the polynomials in a way that will be made formal in Chapter 4 (cf. Section 4.3.1); the general idea being assuming $f(p)/g(p)$

is in $(0,1)$ for all $p \in (0,1)$ and w.l.o.g. that both $f(p)$ and $g(p)$ are positive polynomials on $(0,1)$.

As a consequence of Polya's theorem (presented in Chapter 4), we can increase the degree of the two polynomials so that all the coefficients are positive and both polynomials can be expressed as homogeneous polynomials in the variables p and $1-p$. Therefore, we now write:

$$\frac{f(p)}{g(p)} = \frac{\sum_{k=0}^{n} \binom{n}{k} a_k p^k (1-p)^k}{\sum_{k=0}^{n} \binom{n}{k} b_k p^k (1-p)^k},$$

and the theory we shall develop in Section 4.3.1 ensures $b_k \geq a_k$. Let $M := \max_k \left\{ \binom{n}{k} a_k, \binom{n}{k} b_k \right\}$, clearly:

$$\frac{f(p)}{g(p)} = \frac{\widetilde{f}(p)}{\widetilde{g}(p)} = \frac{\sum_{k=0}^{n} \binom{n}{k} \frac{a_k}{M} p^k (1-p)^k}{\sum_{k=0}^{n} \binom{n}{k} \frac{b_k}{M} p^k (1-p)^k}.$$

Notice that $\frac{a_k}{M} \in [0,1]$ and $(b_k - a_k)/M \in [0,1]$. Finally consider $\widetilde{h}(p) := \widetilde{g}(p) - \widetilde{f}(p)$, thus given by:

$$\widetilde{h}(p) = \sum_{k=0}^{n} \binom{n}{k} \frac{b_k - a_k}{M} p^k (1-p)^k.$$

As shown by Goyal and Sigman [18], it is easy to simulate $\widetilde{f}(p)$ and $\widetilde{h}(p)$ and the required number of tosses is equal to n. Their proposed algorithm proceeds as follows:

1. Let $X_1, \ldots, X_n$ be p-coin tosses and sample $U \sim \text{Unif}(0,1)$,
2. Compute:
$$Y = \sum_{j=0}^{n} \mathbb{I}\{U \leq \alpha_j\} \cdot \mathbb{I}\left\{ \sum_{i=1}^{j} X_i = j \right\},$$
where $\alpha_j = \frac{a_j}{M}$ to simulate $\widetilde{f}(p)$ and it is equal to $\frac{b_j - a_j}{M}$ to simulate $\widetilde{h}(p)$,
3. Return Y.

Finally, we can resort to the division algorithm (cf. Algorithm 9) to obtain a toss of an $f(p)/g(p)$-coin. In this case it can be rewritten as follows.

Algorithm 10 Rational function $f(p)/g(p)$ Bernoulli Factory

1: Let $X \sim \text{Bern}(1/2)$
2: **if** $X = 0$ **then**
3: Let $W \sim \text{Bern}(\widetilde{f}(p))$
4: **if** $W = 1$ **then** stop and output 1 **else** GOTO 1
5: **else**
6: Let $W \sim \text{Bern}(\widetilde{g}(p) - \widetilde{f}(p))$
7: **if** $W = 1$ **then** stop and output 0 **else** GOTO 1
8: **end if**

Notice that each call on line 3 and 6 requires a fixed amount of tosses, equal to n. The probability that the algorithm terminates at each iteration is

$$\frac{1}{2}\widetilde{f}(p) + \frac{1}{2}(\widetilde{g}(p) - \widetilde{f}(p)) = \frac{1}{2}\widetilde{g}(p),$$

therefore the expected number of tosses is:

$$\mathbb{E}[N] = \frac{2n}{\widetilde{g}(p)}.$$

Note that the running time grows to infinite if $\widetilde{g}(p) \to 0$ (or analogously $g(p) \to 0$). The novel technique we will develop in Chapter 4, besides being applicable to dice rather than just coins, will be significantly more efficient and exhibit a running time that does not explode to infinity.

2.8 Discussion

In this chapter we have first recapped the most common approaches taken to design a Bernoulli Factory. We have then shown how a different interpretation of the classic Bernoulli Factory problem, i.e. seeing it as a technique to obtain unbiased estimators, opens up the possibility of using different and novel techniques to devise Bernoulli Factory algorithms. In particular, we have applied the importance sampling and the Russian roulette debiasing techniques to construct Bernoulli Factories for functions that admit power series representation. The proposed algorithms either relax the hypothesis that other works have considered or improves on the expected number of required tosses. A crucial component for these new methodologies is a fast algorithm for linear Bernoulli Factories, and we have provided a comparison of the current state-of-the-art available ones. We believe that the framework we

developed may represent a starting point for whoever needs to design a Bernoulli Factory for a function that arises from a specific problem at hand. We also showed how these techniques are applicable in wider settings, for instance when multiple independent coins are considered.

2.9 Proofs of the results of Chapter 2

Proof of Proposition 2.5

Unbiasedness follows easily:

$$\mathbb{E}\left[\frac{\widehat{\alpha}_K}{\beta_K}\right] = \sum_{k=0}^{\infty} \frac{\mathbb{E}[\widehat{\alpha}_k]}{\beta_k}\beta_k = \sum_{k=0}^{\infty} \alpha_k(p) = f(p),$$

by equation (2.2). Assumptions (iv.a), (v.a) ensure that the procedure's output is always in $[0,1]$, so that Definition 2.4 is satisfied. □

Proof of Proposition 2.7

Unbiasedness is proved in Proposition A.1 of Lyne et al. [35]. Assumption (vi.a) guarantees that the output is in $[0,1]$, so that Definition 2.4 is satisfied. □

Proof of Corollary 2.9

Follows from the discussion of Section 2.4.3. □

Proof of Proposition 2.11

Algorithm 3 produces random variables Y_i having pmf as in eq. (2.3). Following the framework discussed in Section 2.3, notice that:

$$\begin{aligned}\mathbb{P}(Y_k = 1)\prod_{i=0}^{k-1}\mathbb{P}(Y_i = -1) &= p\frac{a_k}{\sum_{j=k}^{\infty} a_j}\frac{\sum_{j=k}^{\infty} a_j}{\sum_{j=k-1}^{\infty} a_j}p \dots \frac{\sum_{j=2}^{\infty} a_j}{\sum_{j=1}^{\infty} a_j}p\left(1-\frac{a_0}{M}\right) \\ &= \frac{a_k p^k}{\sum_{j=1}^{\infty} a_j}\left(1-\frac{a_0}{M}\right) = \frac{a_k p^k}{M-a_0}\frac{M-a_0}{M} = \frac{a_k p^k}{M},\end{aligned}$$

thus proving that $\widetilde{f}(p) := \frac{1}{M}\sum_{k=0}^{\infty} a_k p^k = \sum_{k=0}^{\infty}\mathbb{P}(Y_k = 1)\prod_{i=0}^{k-1}\mathbb{P}(Y_i = -1)$, as desired.

The number N of required p-tosses equals the number of iterations required (starting from 0), and therefore satisfies:

$$\begin{aligned}\mathbb{P}(N > n) = \prod_{i=0}^{n}\mathbb{P}(Y_i = -1) &= \underbrace{\left(1-\frac{a_0}{M}\right)\frac{\sum_{j=2}^{\infty} a_j}{\sum_{j=1}^{\infty} a_j}}_{\frac{\sum_{j=2}^{\infty} a_j}{M}} p\frac{\sum_{j=3}^{\infty} a_j}{\sum_{j=2}^{\infty} a_j}p \dots \frac{\sum_{j=n+1}^{\infty} a_j}{\sum_{j=n}^{\infty} a_j}p \\ &= \frac{\sum_{j=n+1}^{\infty} a_j}{M}p^n,\end{aligned}$$

so that it follows

$$\begin{aligned}\mathbb{E}[N] &= \sum_{n=0}^{\infty} \frac{p^n}{M} \sum_{j=n+1}^{\infty} a_j = \sum_{j=1}^{\infty} \frac{a_j}{M} \sum_{n=0}^{j-1} p^n = \sum_{j=1}^{\infty} \frac{a_j}{M} \frac{1-p^j}{1-p} \\ &= \frac{1}{M(1-p)} \left(\sum_{j=1}^{\infty} a_j - \sum_{j=1}^{\infty} a_j p^j \right) = \frac{1}{M(1-p)} \left(M - a_0 - f(p) + a_0 \right) \\ &= \frac{M - f(p)}{M(1-p)}.\end{aligned}$$

□

Proof of Proposition 2.12

We show that all the conditions of Proposition 2.5 are satisfied. We set $\beta_k = a_k b^k / M$, so that conditions (i.a), (ii) are met. The estimator of $\alpha_k := a_k p^k$ is constructed as:

$$\widehat{\alpha}_k = a_k \prod_{i=1}^{k} \widehat{p}_i,$$

as the estimates $\widehat{p}_1, \ldots, \widehat{p}_K$ are independent and unbiased estimators of p it follows that conditions (iii), (iv.a) are also satisfied. Finally, as we assume that the estimators $\widehat{p}_i$ are smaller or equal than b a.s., condition (v.a) is also easily verified. The number of p-coin tosses required is at most K, as we can prematurely stop the algorithm as soon as tails is observed. □

Proof of Corollary 2.13

If $b \leq 1$, we can convert unbiased estimates of p to be in $\{0, 1\}$ by sampling $U_i \sim \text{Unif}(0, 1)$ and checking $\mathbb{I}\{U_i < \widehat{p}_i\}$. Therefore, there is a positive probability that $\widehat{p}_i = 0$ and the algorithm can be prematurely stopped, reducing the running time. This is effectively as setting $b = 1$ in the value of M of Proposition 2.12.

Given $K > 0$, i.e. the number of iterations required for the algorithm to termi-

nate, the number N of required p-coin tosses follows the following distribution:

$$\mathbb{P}(N = n|K = k, k > 0) = \begin{cases} p^{n-1}(1-p) & \text{if } 0 < n < k \\ p^{n-1} & \text{if } n = k \\ 0 & \text{if } n \leq 0 \text{ or } n > k \end{cases},$$

$$\mathbb{P}(N > n|K = k, k > 0) = \begin{cases} 1 & \text{if } n < 0 \\ p^n & \text{if } 0 \leq n < k \\ 0 & \text{if } n \geq k \end{cases}. \tag{2.6}$$

Recall that $\mathbb{P}(K = k) = a_k/M$, so that for $n \geq 0$:

$$\begin{aligned} \mathbb{P}(N > n) &= \sum_{k=0}^{\infty} \mathbb{P}(N > n|K = k)\mathbb{P}(K = k) = \frac{1}{M}\sum_{k=1}^{\infty} a_k \mathbb{P}(N > n|K = k) \\ &= \frac{p^n}{M}\sum_{k=n+1}^{\infty} a_k. \end{aligned}$$

Finally, noticing that $M := \sum_{k=0}^{\infty} a_k$, we obtain:

$$\begin{aligned} \mathbb{E}[N] &= \frac{1}{M}\sum_{n=0}^{\infty}\sum_{k=n+1}^{\infty} a_k p^n = \frac{1}{M}\sum_{k=1}^{\infty}\sum_{n=0}^{k-1} a_k p^n = \frac{1}{M}\sum_{k=1}^{\infty} a_k \cdot \frac{1-p^k}{1-p} \\ &= \frac{1}{M(1-p)}\left(\sum_{k=1}^{\infty} a_k - \sum_{k=1}^{\infty} a_k p^k\right) = \frac{1}{M(1-p)}\left(M - a_0 - f(p) + a_0\right) \\ &= \frac{M - f(p)}{M(1-p)}. \end{aligned}$$

□

Proof of Proposition 2.14

Under the framework of Section 2.4.2, let $\alpha_k(p) := \frac{a_k}{M}p^k$, $\widehat{\alpha}_k := \frac{a_k}{M}\prod_{i=1}^{k}\widehat{p}_i^k$, and $\omega_k = b^k$ (which is equivalent to requiring $K \sim \text{Geom}(1-b)$). Conditions (i.b), (ii), (iii), (iv.a) are clearly satisfied. Note that:

$$\sup_k \sum_{i=0}^{k} \frac{\widehat{\alpha}_i}{\omega_i} = \sup_k \sum_{i=0}^{k} \frac{a_i \prod_{j=1}^{i} \widehat{p}^j}{Mb^i} \leq \sup_k \sum_{i=0}^{k} \frac{a_i b^i}{Mb^i} = \frac{1}{M}\sum_{i=0}^{\infty} a_i = 1,$$

so that also condition (vi.a) is satisfied. Therefore, Algorithm 5 is a valid Bernoulli Factory for $\widetilde{f}(p)$ by Proposition 2.7. To conclude, note that the number of unbiased

estimates of p required is at most equal to K, as we can prematurely stop the algorithm as soon as tails is observed. □

Proof of Proposition 2.18

Algorithm 6 produces random variables Y_i having pmf as in eq. (2.4), it is enough to notice that:

$$\mathbb{P}(Y_k = 1) = \mathbb{P}(X = 1, W = 0) = \frac{a_{2k}}{a_{2k-1}} p \left(1 - \frac{a_{2k+1}}{a_{2k}} p\right) = \frac{p}{a_{2k-1}} (a_{2k} - a_{2k+1} p).$$

According to the framework developed in Section 2.3 we have:

$$\begin{aligned}\mathbb{P}(Y_k = 1) \prod_{i=0}^{k-1} \mathbb{P}(Y_i = -1) &= \frac{p}{a_{2k-1}} (a_{2k} - a_{2k+1} p) \frac{a_{2k-1}}{a_{2k-3}} p^2 \frac{a_{2k-3}}{a_{2k-5}} p^2 \dots \frac{a_3}{a_1} p^2 \frac{a_1}{a_0} p \\ &= \frac{a_{2k}}{a_0} p^{2k} - \frac{a_{2k+1}}{a_0} p^{2k+1},\end{aligned}$$

so that $\widetilde{f}(p) := \frac{1}{a_0} \sum_{k=0}^{\infty} (-1)^k a_k p^k = \sum_{k=0}^{\infty} \mathbb{P}(Y_k = 1) \prod_{i=0}^{k-1} \mathbb{P}(Y_i = -1)$, as desired.

Lt K be the number of iterations required for the algorithm to terminate. Then $\mathbb{P}(K = 0) = 1 - \frac{a_1}{a_0} p$ and for $k > 0$:

$$\begin{aligned}\mathbb{P}(K = k) = \mathbb{P}(Y_k \neq -1) \prod_{i=0}^{k-1} \mathbb{P}(Y_i = -1) &= \left(1 - \frac{a_{2k+1}}{a_{2k-1}} p^2\right) \frac{a_1}{a_0} p \frac{a_3}{a_1} p^2 \frac{a_5}{a_3} p^2 \dots \frac{a_{2k-1}}{a_{2k-3}} p^2 \\ &= \left(1 - \frac{a_{2k+1}}{a_{2k-1}} p^2\right) \frac{a_{2k-1}}{a_0} p^{2k-1} = \frac{a_{2k-1} p^{2k-1} - a_{2k+1} p^{2k+1}}{a_0} \\ &= \frac{p^{2k-1}}{a_0} (a_{2k-1} - a_{2k+1} p^2)\end{aligned}$$

Let N be the number of tosses of the p-coin required. At the 0^{th} iteration, 1 toss is required, while for each of the successive iterations 2 tosses are required. Therefore:

$$\mathbb{P}(N > n | K = k) = \begin{cases} 1 & \text{if } n < 2k + 1 \\ 0 & \text{otherwise} \end{cases}$$

so that $\mathbb{P}(N > 0) = 1$ and for $n > 0$:

$$\mathbb{P}(N > n) = \sum_{k=1}^{\infty} \mathbb{P}(N > n | K = k) \mathbb{P}(K = k) = \frac{1}{a_0} \sum_{k=\lceil n/2 \rceil}^{\infty} p^{2k-1} (a_{2k-1} - a_{2k+1} p^2).$$

Finally:

$$\begin{aligned}
\mathbb{E}[N] &= \sum_{n=0}^{\infty} \mathbb{P}(N > n) = 1 + \sum_{n=1}^{\infty} \mathbb{P}(N > n) \\
&= 1 + \frac{1}{a_0} \sum_{n=1}^{\infty} \sum_{k=\lceil n/2 \rceil}^{\infty} p^{2k-1}(a_{2k-1} - a_{2k+1}p^2) \\
&= 1 + \frac{1}{a_0} \sum_{k=1}^{\infty} \sum_{n=1}^{2k} p^{2k-1}(a_{2k-1} - a_{2k+1}p^2) \\
&= 1 + \frac{2}{a_0} \sum_{k=1}^{\infty} kp^{2k-1}(a_{2k-1} - a_{2k+1}p^2) \\
&= 1 + \frac{2}{a_0} \left[a_1 p - a_3 p^3 + 2a_3 p^3 - 2a_5 p^5 + 3a_5 p^5 - 3a_7 p^7 + 4a_7 p^7 - 4a_9 p^9 + \ldots \right] \\
&= 1 + \frac{2}{a_0} \sum_{k=1}^{\infty} a_{2k-1} p^{2k-1}.
\end{aligned}$$

□

Proof of Proposition 2.20

We shall check that conditions (i.a), (ii), (iii), (iv.a), (v.a) are satisfied, so that Proposition 2.5 holds. Notice that here $\alpha_k(p)$ is defined to be:

$$\alpha_k(p) := \frac{1}{M}(a_{2k}p^{2k} - a_{2k+1}p^{2k+1}).$$

Given independent unbiased estimates $\widehat{p}_1, \widehat{p}_2, \ldots$ of p, the algorithm considers unbiased estimates of $\alpha_k(p)$ and proposal probabilities β_k defined as:

$$\widehat{\alpha}_k = \frac{1}{M}\left(a_{2k} \prod_{i=1}^{2k} \widehat{p}_i - a_{2k+1} \prod_{i=1}^{2k+1} \widehat{p}_i \right), \qquad \beta_k = \frac{1}{M}\left(a_{2k} b^{2k} - a_{2k+1} a^{2k+1} \right).$$

Condition (i.a) is guaranteed by the assumption $\frac{a_{2k}}{a_{2k+1}} \geq \frac{a^{2k+1}}{b^{2k}}$, while conditions (ii), (iii) are clearly satisfied. Condition (iv.a) is equivalent to requiring $a_{2k} - a_{2k+1}\widehat{p}_{2k+1} \geq 0$ for all k, which is the assumption we are making $b \leq \max_k \frac{a_{2k}}{a_{2k+1}}$. Finally, condition (v.a) is also satisfied:

$$\sup_k \frac{\widehat{\alpha}_k}{\beta_k} = \sup_k \frac{a_{2k} \prod_{i=1}^{2k} \widehat{p}_i - a_{2k+1} \prod_{i=1}^{2k+1} \widehat{p}_i}{a_{2k} b^{2k} - a_{2k+1} a^{2k+1}} \leq \frac{a_{2k} \prod_{i=1}^{2k} b - a_{2k+1} \prod_{i=1}^{2k+1} a}{a_{2k} b^{2k} - a_{2k+1} a^{2k+1}} = 1.$$

To conclude, note that the number of tosses N required by the algorithm is at

most equal to $2K+1$ (as we can prematurely stop the algorithm as soon as tails is observed), so that:

$$\mathbb{P}(N=n) \leq \begin{cases} \frac{1}{M}\left(a_{n-1}b^{n-1} - a_n a^n\right) & \text{if } n \text{ is odd} \\ 0 & \text{otherwise.} \end{cases}$$

□

Proof of Corollary 2.22

This scenario corresponds to the classic Bernoulli Factory case, where $a=0$ and $b=1$. The assumptions guarantee that all conditions of Proposition 2.20 are satisfied, so that Algorithm 7 is a valid Bernoulli Factory for $\widetilde{f}(p) := f(p)/M$.

Given K, the algorithm requires at most $2K+1$ tosses, and it can stop as soon as one the toss results in a tail. Therefore, following a similar reasoning as in the proof of Corollary 2.13, we have:

$$\mathbb{P}(N>n|K=k) = \begin{cases} 1 & \text{if } n<0 \\ 0 & \text{if } 0 \leq n \geq 2k+1 \\ p^n & \text{if } n < 2k+1 \end{cases}$$

and therefore:

$$\mathbb{P}(N>n) = \sum_{k=0}^{\infty} \mathbb{P}(N>n|K=k)\frac{a_{2k}}{M} = \frac{p^n}{M}\sum_{k=\lceil n/2 \rceil}^{\infty} a_{2k}.$$

Finally:

$$\mathbb{E}[N] = \frac{1}{M}\sum_{n=0}^{\infty}\sum_{k=\lceil n/2 \rceil}^{\infty} a_{2k}p^n = \frac{1}{M}\sum_{k=0}^{\infty}\sum_{n=0}^{2k} a_{2k}p^n = \frac{1}{M(1-p)}\sum_{k=0}^{\infty} a_{2k}(1-p^{2k+1}).$$

□

Proof of Proposition 2.24

We shall check that conditions (i.b), (ii), (iii), and (vi.a) are satisfied, so that by Proposition 2.7 we reach the required result. In this case, $\alpha_k(p)$ is defined as:

$$\alpha_k(p) := (-1)^k \frac{a_k}{a_0}p^k,$$

and given unbiased estimates $\widehat{p}_1, \widehat{p}_2, \ldots$ of p the algorithm considers unbiased estimates of $\alpha_k(p)$ and sets the sequence $\{\omega_k\}_{k\in\mathbb{N}}$ as:

$$\widehat{\alpha_k} := (-1)^k \frac{a_k}{a_0} \prod_{i=1}^{k} \widehat{p}_i, \qquad \omega_k := \frac{a_k}{a_0},$$

or, equivalently, $\beta_k := \omega_k - \omega_{k+1} = \frac{a_k - a_{k+1}}{a_0}$.

The assumption of decreasing coefficients along with $a_k \to 0$ as $k \to \infty$ ensures that condition (i.b) is satisfied. Conditions (ii) and (iii) are clearly satisfied. Notice that $\widehat{\alpha}_k/\omega_k = (-1)^k \prod_{i=1}^{k} \widehat{p}_i$, and that we can assume w.l.o.g. $\widehat{p}_i \in \{0, 1\}$

$$\sum_{k=0}^{n} \frac{1}{\omega_k} \widehat{\alpha_k} = \sum_{k=0}^{\tau} (-1)^k,$$

where τ is the minimum i such that $\widehat{p}_i = 0$ or it is equal to n if this does not happen. Then clearly $\sum_{k=0}^{\tau} (-1)^k \in \{0, 1\}$ and condition (vi.a) is also satisfied.

The algorithm requires up to K tosses, as we can stop as soon as tails is observed. For $K > 0$, $N|K$ follows the distribution detailed in eq. (2.6). Since $\mathbb{P}(K = k) = (a_k - a_{k+1})/a_0$, we obtain for $n \geq 0$:

$$\begin{aligned}\mathbb{P}(N > n) &= \sum_{k=0}^{\infty} \mathbb{P}(N > n | K = k)\mathbb{P}(K = k) = \frac{1}{a_0} \sum_{k=1}^{\infty} (a_k - a_{k+1})\mathbb{P}(N > n | K = k) \\ &= \frac{p^n}{a_0} \sum_{k=n+1}^{\infty} (a_k - a_{k+1}) = \frac{a_{n+1}}{a_0} p^n,\end{aligned}$$

and finally:

$$\mathbb{E}[N] = \frac{1}{a_0} \sum_{n=0}^{\infty} a_{n+1} p^n.$$

□

2.10 Appendix

Comparison of Huber's algorithms for linear Bernoulli Factory

	ϵ =0.99		
	$\widehat{N}_{2016}$	$\widehat{N}_{2017}$	$\widehat{N}_{2019}$
$M = 2$	$8.11 \cdot 10^0$ (5)	$1.20 \cdot 10^1$ (11)	$\mathbf{6.00 \cdot 10^0}$ (5)
$M = 5$	$2.96 \cdot 10^1$ (19)	$3.00 \cdot 10^1$ (28)	$\mathbf{1.49 \cdot 10^1}$ (13)
$M = 10$	$6.51 \cdot 10^1$ (41)	$6.01 \cdot 10^1$ (56)	$\mathbf{3.00 \cdot 10^1}$ (26)
$M = 50$	$3.54 \cdot 10^2$ (221)	$3.00 \cdot 10^2$ (280)	$\mathbf{1.50 \cdot 10^2}$ (133)
$M = 100$	$7.06 \cdot 10^2$ (444)	$6.00 \cdot 10^2$ (559)	$\mathbf{2.99 \cdot 10^2}$ (265)
$M = 500$	$3.60 \cdot 10^3$ (2265)	$3.01 \cdot 10^3$ (2815)	$\mathbf{1.50 \cdot 10^3}$ (1336)
$M = 1000$	$7.15 \cdot 10^3$ (4511)	$6.03 \cdot 10^3$ (5605)	$\mathbf{3.00 \cdot 10^3}$ (2673)
	ϵ =0.90		
	$\widehat{N}_{2016}$	$\widehat{N}_{2017}$	$\widehat{N}_{2019}$
$M = 2$	$8.17 \cdot 10^0$ (6)	$1.21 \cdot 10^1$ (11)	$\mathbf{5.94 \cdot 10^0}$ (5)
$M = 5$	$2.97 \cdot 10^1$ (21)	$3.02 \cdot 10^1$ (27)	$\mathbf{1.47 \cdot 10^1}$ (12)
$M = 10$	$6.60 \cdot 10^1$ (47)	$6.05 \cdot 10^1$ (55)	$\mathbf{2.94 \cdot 10^1}$ (25)
$M = 50$	$3.55 \cdot 10^2$ (253)	$3.01 \cdot 10^2$ (278)	$\mathbf{1.47 \cdot 10^2}$ (125)
$M = 100$	$7.12 \cdot 10^2$ (511)	$6.02 \cdot 10^2$ (554)	$\mathbf{2.96 \cdot 10^2}$ (251)
$M = 500$	$3.59 \cdot 10^3$ (2574)	$3.03 \cdot 10^3$ (2796)	$\mathbf{1.48 \cdot 10^3}$ (1266)
$M = 1000$	$7.20 \cdot 10^3$ (5155)	$6.03 \cdot 10^3$ (5566)	$\mathbf{2.97 \cdot 10^3}$ (2522)
	ϵ =0.75		
	$\widehat{N}_{2016}$	$\widehat{N}_{2017}$	$\widehat{N}_{2019}$
$M = 2$	$8.61 \cdot 10^0$ (6)	$1.42 \cdot 10^1$ (12)	$\mathbf{7.93 \cdot 10^0}$ (6)
$M = 5$	$3.14 \cdot 10^1$ (23)	$3.55 \cdot 10^1$ (31)	$\mathbf{1.99 \cdot 10^1}$ (15)
$M = 10$	$6.95 \cdot 10^1$ (51)	$7.09 \cdot 10^1$ (63)	$\mathbf{3.94 \cdot 10^1}$ (31)
$M = 50$	$3.71 \cdot 10^2$ (278)	$3.55 \cdot 10^2$ (318)	$\mathbf{1.99 \cdot 10^2}$(159)
$M = 100$	$7.61 \cdot 10^2$ (560)	$7.07 \cdot 10^2$ (637)	$\mathbf{4.03 \cdot 10^2}$ (318)
$M = 500$	$3.84 \cdot 10^3$ (2836)	$3.56 \cdot 10^3$ (3208)	$\mathbf{1.98 \cdot 10^3}$ (1602)
$M = 1000$	$7.63 \cdot 10^3$ (5700)	$7.10 \cdot 10^3$ (6380)	$\mathbf{3.98 \cdot 10^3}$ (3211)

	ϵ =0.50		
	$\widehat{N}_{2016}$	$\widehat{N}_{2017}$	$\widehat{N}_{2019}$
$M = 2$	$\mathbf{1.12 \cdot 10^1}$ (6)	$2.08 \cdot 10^1$ (13)	$1.46 \cdot 10^1$ (8)
$M = 5$	$4.18 \cdot 10^1$ (24)	$5.20 \cdot 10^1$ (34)	$\mathbf{3.54 \cdot 10^1}$ (21)
$M = 10$	$9.32 \cdot 10^1$ (55)	$1.04 \cdot 10^2$ (71)	$\mathbf{7.10 \cdot 10^1}$ (45)
$M = 50$	$5.02 \cdot 10^2$ (302)	$5.19 \cdot 10^2$ (357)	$\mathbf{3.54 \cdot 10^2}$ (224)
$M = 100$	$1.02 \cdot 10^3$ (611)	$1.04 \cdot 10^3$ (718)	$\mathbf{7.28 \cdot 10^2}$ (453)
$M = 500$	$5.20 \cdot 10^3$ (3126)	$5.21 \cdot 10^3$ (3577)	$\mathbf{3.75 \cdot 10^3}$ (2259)
$M = 1000$	$1.02 \cdot 10^4$ (6060)	$1.04 \cdot 10^4$ (7179)	$\mathbf{7.24 \cdot 10^3}$ (4560)
	ϵ =0.25		
	$\widehat{N}_{2016}$	$\widehat{N}_{2017}$	$\widehat{N}_{2019}$
$M = 2$	$\mathbf{2.05 \cdot 10^1}$ (3)	$4.09 \cdot 10^1$ (5)	$3.36 \cdot 10^1$ (5)
$M = 5$	$\mathbf{8.05 \cdot 10^1}$ (9)	$1.03 \cdot 10^2$ (11)	$8.11 \cdot 10^1$ (11)
$M = 10$	$1.82 \cdot 10^2$ (18)	$2.04 \cdot 10^2$ (22)	$\mathbf{1.59 \cdot 10^2}$ (20)
$M = 50$	$9.67 \cdot 10^2$ (96)	$1.02 \cdot 10^3$ (109)	$\mathbf{8.17 \cdot 10^2}$ (98)
$M = 100$	$1.99 \cdot 10^3$ (198)	$2.05 \cdot 10^3$ (217)	$\mathbf{1.62 \cdot 10^3}$ (199)
$M = 500$	$9.80 \cdot 10^3$ (975)	$1.03 \cdot 10^4$ (1092)	$\mathbf{8.21 \cdot 10^3}$ (989)
$M = 1000$	$1.98 \cdot 10^4$ (1964)	$2.03 \cdot 10^4$ (2147)	$\mathbf{1.57 \cdot 10^4}$ (1950)
	ϵ =0.10		
	$\widehat{N}_{2016}$	$\widehat{N}_{2017}$	$\widehat{N}_{2019}$
$M = 2$	$\mathbf{5.00 \cdot 10^1}$ (2)	$1.02 \cdot 10^2$ (3)	$8.30 \cdot 10^1$ (3)
$M = 5$	$\mathbf{2.00 \cdot 10^2}$ (6)	$2.59 \cdot 10^2$ (8)	$2.07 \cdot 10^2$ (7)
$M = 10$	$4.35 \cdot 10^2$ (13)	$5.13 \cdot 10^2$ (15)	$\mathbf{4.29 \cdot 10^2}$ (14)
$M = 50$	$\mathbf{2.42 \cdot 10^3}$ (66)	$2.58 \cdot 10^3$ (70)	$2.77 \cdot 10^3$ (68)
$M = 100$	$4.75 \cdot 10^3$ (133)	$5.07 \cdot 10^3$ (140)	$\mathbf{4.17 \cdot 10^3}$ (136)
$M = 500$	$2.47 \cdot 10^4$ (675)	$2.60 \cdot 10^4$ (706)	$\mathbf{2.26 \cdot 10^4}$ (675)
$M = 1000$	$4.80 \cdot 10^4$ (1339)	$5.17 \cdot 10^4$ (1390)	$\mathbf{4.03 \cdot 10^4}$ (1348)
	ϵ =0.01		
	$\widehat{N}_{2016}$	$\widehat{N}_{2017}$	$\widehat{N}_{2019}$
$M = 2$	$\mathbf{5.24 \cdot 10^2}$ (2)	$1.00 \cdot 10^3$ (3)	$8.24 \cdot 10^2$ (3)
$M = 5$	$\mathbf{1.90 \cdot 10^3}$ (5)	$2.57 \cdot 10^3$ (6)	$1.95 \cdot 10^3$ (6)
$M = 10$	$\mathbf{4.24 \cdot 10^3}$ (11)	$4.99 \cdot 10^3$ (12)	$4.57 \cdot 10^3$ (12)
$M = 50$	$2.29 \cdot 10^4$ (57)	$2.56 \cdot 10^4$ (57)	$\mathbf{2.12 \cdot 10^4}$ (57)
$M = 100$	$\mathbf{4.59 \cdot 10^4}$ (114)	$5.13 \cdot 10^4$ (115)	$4.75 \cdot 10^4$ (113)
$M = 500$	$2.38 \cdot 10^5$ (565)	$2.59 \cdot 10^5$ (573)	$\mathbf{2.00 \cdot 10^5}$ (571)
$M = 1000$	$4.57 \cdot 10^5$ (1128)	$4.88 \cdot 10^5$ (1138)	$\mathbf{4.21 \cdot 10^5}$ (1142)

	ϵ =0.001		
	$\widehat{N}_{2016}$	$\widehat{N}_{2017}$	$\widehat{N}_{2019}$
$M = 2$	$\mathbf{4.96 \cdot 10^3}$ (2)	$9.29 \cdot 10^3$ (3)	$9.54 \cdot 10^3$ (3)
$M = 5$	$\mathbf{1.63 \cdot 10^4}$ (5)	$2.73 \cdot 10^4$ (6)	$2.86 \cdot 10^4$ (6)
$M = 10$	$\mathbf{3.30 \cdot 10^4}$ (11)	$5.20 \cdot 10^4$ (12)	$8.76 \cdot 10^4$ (12)
$M = 50$	$2.09 \cdot 10^5$ (55)	$2.12 \cdot 10^5$ (57)	$\mathbf{2.06 \cdot 10^5}$ (56)
$M = 100$	$4.59 \cdot 10^5$ (112)	$4.97 \cdot 10^5$ (114)	$\mathbf{2.57 \cdot 10^5}$ (112)
$M = 500$	$3.12 \cdot 10^6$ (564)	$2.56 \cdot 10^6$ (556)	$\mathbf{9.80 \cdot 10^5}$ (560)
$M = 1000$	$4.64 \cdot 10^6$ (1119)	$5.67 \cdot 10^6$ (1121)	$\mathbf{1.03 \cdot 10^6}$ (1125)

Table 2: Comparison of three algorithms for linear Bernoulli Factories proposed by Huber (2016, 2017, 2019) [23, 24, 25] for different choices of the constant M and bound ϵ. The true value of p is set equal to $(1 - \epsilon)/M$. The table presents the empirical average of p-coin tosses required over 100,000 replications, while in bracket the empirical median is given. Bold text represents the best algorithm in terms of expected number of tosses.

3

From a Bernoulli Factory to a Dice Enterprise

3.1 Introduction

In this chapter we propose to formally study a multivariate extension of the Bernoulli Factory problem so to consider categorical random variables, named *Dice Enterprise*. More specifically, given a loaded die with an arbitrary number of faces we study the existence of algorithms that produce rolls of a dice such that the probability of rolling each face is a fixed function of the probabilities associated with the given die. Section 3.2 generalises the work of Łatuszyński et al. [32] and looks at the generic problem of simulating from categorical distributions of unknown parameter. We provide implementable algorithms that make use of suitably constructed upper and lower bounds of the parameter. Section 3.3 formally introduces the Dice Enterprise problem as a multivariate extension of the Bernoulli Factory. It makes use of the framework developed in the previous section to show that for a Dice Enterprise to exist for a function $f(\boldsymbol{p})$, it must be possible to approximate f via suitable polynomials. We also prove for which class of functions a Dice Enterprise can be constructed, extending Theorem 1.2. Section 3.4 looks at providing efficient Dice Enterprise type of algorithms. We give a novel algorithm for functions that admit specific power series expansion, extending and improving on previous results [38, 43]. We show that only analytic functions can have a fast simulation, in a sense that will be made more formal later on, extending the results of Nacu and Peres [43] (Theorem 2). Finally, Section 3.5 studies the related problem of having access to multiple independent coins of unknown bias rather than a die. We also consider the case where an infinite number of coins or die faces are given and give initial insights into how a Dice Enterprise for this case may work. Code to reproduce examples is made available at `https://github.com/giuliomorina/DiceEnterpriseSeries`.

3.2 Simulation of Categorical Random Variables of Unknown Parameter

We now focus on the problem of simulating from a categorical distribution of unknown parameter $\boldsymbol{p} \in \Delta^m$ - extending the work of Łatuszyński et al. [32] which focused solely on unidimensional problems. To this end, we first notice that knowing $\boldsymbol{p}$ is not needed: it suffices to have access to an unbiased estimator $\widehat{\boldsymbol{p}}$ which is in $\bar{\Delta}^m$ almost surely. We then consider the case where we have access to a monotonic sequence of estimators converging to $\boldsymbol{p}$ and provide an algorithm to sample from the corresponding categorical distribution, also computing its running time. Importantly, the sequence of estimators does not need to be deterministic. Finally, we further relax the assumption of monotonicity and just require for the sequence to be a reverse time supermartingale, in a sense that will be made precise later. The algorithms presented in this section will be a fundamental building block to the theory of Dice Enterprise that shall be developed in the sequel.

Throughout this section we will assume to have access to a generator of uniform random variables U, to evaluate events of the form $\{U < u\}$. This is usually common in practice, but under the assumption $\boldsymbol{p} \in \Delta^m$, we have seen in Section 1.4 how we can achieve this even if the only source of randomness is given by means of a black-box outputting draws from a categorical distribution of unknown parameter.

Equivalence sampling/unbiased

A first crucial observation presented in the following lemma, which can be seen as a generalisation of Lemma 2.3, is that obtaining rolls of a $\boldsymbol{p}$-die is equivalent to constructing an unbiased estimator of $\boldsymbol{p}$ which is in $\bar{\Delta}^m$ almost surely. The proof readily provides an algorithm to perform the sampling given an estimator satisfying such properties.

Lemma 3.1. *Sampling random variables having a Categorical distribution of parameter $\boldsymbol{p}$ is equivalent to constructing an unbiased estimator $\widehat{\boldsymbol{p}}$ of $\boldsymbol{p}$ taking values in $\bar{\Delta}^m$ with probability 1.*

Proof. $\Rightarrow$ Given $X \sim \boldsymbol{p}$ let $\widehat{\boldsymbol{p}}$ be a vector of zeros with a 1 in the X^{th} position. It follows that $\widehat{\boldsymbol{p}}$ is an unbiased estimator of $\boldsymbol{p}$.

$\Leftarrow$ Let $\widehat{\boldsymbol{s}} = \left(\widehat{p}_0, \widehat{p}_0 + \widehat{p}_1, \ldots, \sum_{i=0}^{m-1} \widehat{p}_i, 1\right)$ and draw $U \sim \text{Unif}(0,1)$. Consider setting:

$$X = \begin{cases} 0 & \text{if } U \leq \widehat{s}_0 \\ 1 & \text{if } \widehat{s}_0 < U \leq \widehat{s}_1 \\ & \ldots \\ m-1 & \text{if } \widehat{s}_{m-2} < U \leq \widehat{s}_{m-1} \\ m & \text{if } U > \widehat{s}_{m-1} \end{cases}$$

Then for any $k \in \{0, \ldots, m\}$,

$$\mathbb{P}(X = k) = \mathbb{E}[\mathbb{E}[\mathbb{I}(X = k)|\widehat{\boldsymbol{s}}]] = \mathbb{E}\left[\mathbb{I}(\widehat{s}_{k-1} < U \leq \widehat{s}_k)\right] = \mathbb{E}[\widehat{s}_k - \widehat{s}_{k-1}] = p_k,$$

where $\widehat{s}_{-1} = 0$. □

Deterministic bounds

Let $\boldsymbol{l}^{(n)} := \left(l_0^{(n)}, \ldots, l_m^{(n)}\right)$ be an increasing sequence of lower bounds of $\boldsymbol{p}$ such that the following holds for all $n \in \mathbb{N}$:

(i) $\boldsymbol{l}^{(n)} \in [0,1]^{m+1}$

(ii) $\boldsymbol{l}^{(n)} \leq \boldsymbol{l}^{(n+1)}$

(iii) $\lim_{n\to\infty} \boldsymbol{l}^{(n)} = \boldsymbol{p}$

Note that the three conditions imply $\left\|\boldsymbol{l}^{(n)}\right\|_1 \leq 1, \forall n \in \mathbb{N}$. Algorithm 11 produces rolls of a $\boldsymbol{p}$-die given such a sequence, as proved in Lemma 3.4.

Remark 3.2. If a decreasing sequence $\boldsymbol{u}^{(n)}$ is given, rather than an increasing one, we can construct $\boldsymbol{l}^{(n)}$ by setting $l_i^{(n)} = 1 - \sum_{j\neq i} u_i^{(n)}$. Conditions (i)-(iii) are easily verified assuming: $\boldsymbol{u}^{(n)} \in [0,1]^{m+1}$, $\boldsymbol{u}^{(n)} \geq \boldsymbol{u}^{(n+1)}$, and $\lim_{n\to\infty} \boldsymbol{u}^{(n)} = \boldsymbol{p}$.

Remark 3.3. In the case of Bernoulli distributions, Łatuszyński et al. [32] considers lower bounds $l^{(n)}$ and upper bounds $u^{(n)}$ for the probability p of tossing heads. Here, we slightly change perspective and consider providing lower bounds for the probabilities of tossing tails and heads. The two approaches are equivalent: it is enough to set $l_0^{(n)} = 1 - u^{(n)}$ and $l_1^{(n)} = l^{(n)}$.

Algorithm 11 Deterministic bounds

1: Set $\boldsymbol{l}^{(0)} := \mathbf{0}$, $n := 1$, and simulate $U \sim U(0,1)$
2: Compute $\boldsymbol{l}^{(n)}$
3: **if** $U < \left\|\boldsymbol{l}^{(n)}\right\|_1$ **then**
4: **return** $i \in \{0, \dots, m\}$ such that

$$\left\|\boldsymbol{l}^{(n-1)}\right\|_1 + \sum_{j=0}^{i-1} \left(l_j^{(n)} - l_j^{(n-1)}\right) \leq U < \left\|\boldsymbol{l}^{(n-1)}\right\|_1 + \sum_{j=0}^{i} \left(l_j^{(n)} - l_j^{(n-1)}\right)$$

5: **else**
6: Set $n = n + 1$ and GOTO 2
7: **end if**

Lemma 3.4. *Assume (i), (ii), (iii). Then Algorithm 11 outputs a roll of a $\boldsymbol{p}$-die and the probability that it needs $N > n$ iterations equals*

$$\mathbb{P}(N > n) = 1 - \left\|\boldsymbol{l}^{(n)}\right\|_1.$$

Proof. Special case of Lemma 3.5. □

Random bounds

We now generalise the previous construction and take into account randomised bounds, that is estimators $\boldsymbol{L}^{(n)}$ of $\boldsymbol{l}^{(n)}$. We rephrase the previous conditions as:

(I) $\boldsymbol{L}^{(n)} \in [0,1]^{m+1}$ a.s.

(II) $\boldsymbol{L}^{(n)} \leq \boldsymbol{L}^{(n+1)}$ a.s.

(III) $\lim_{n\to\infty} \mathbb{E}\left[\boldsymbol{L}^{(n)}\right] := \boldsymbol{l}^{(n)} \to \boldsymbol{p}$

Following the notation of Łatuszyński et al. [32], let

$$\mathcal{F}_0 = \{\emptyset, \Omega\}, \qquad \mathcal{F}_n = \sigma\left\{\boldsymbol{L}^{(\boldsymbol{n})}\right\}, \qquad \mathcal{F}_{k:n} = \sigma\left\{\mathcal{F}_k, \mathcal{F}_{k+1}, \dots, \mathcal{F}_n\right\} \text{ for } k \leq n,$$

and consider the following algorithm, where recall that $\|\cdot\|_1$ denotes the 1-norm.

Algorithm 12 Random bounds

1: Set $\boldsymbol{L}^{(0)} := \mathbf{0}$, $n := 1$, and simulate $U \sim U(0,1)$
2: Obtain $\boldsymbol{L}^{(n)}$ given $\mathcal{F}_{0:(n-1)}$
3: **if** $U < \left\|\boldsymbol{L}^{(n)}\right\|_1$ **then**
4: **return** $i \in \{0,\dots,m\}$ such that

$$\left\|\boldsymbol{L}^{(n-1)}\right\|_1 + \sum_{j=0}^{i-1}\left(L_j^{(n)} - L_j^{(n-1)}\right) \leq U < \left\|\boldsymbol{L}^{(n-1)}\right\|_1 + \sum_{j=0}^{i}\left(L_j^{(n)} - L_j^{(n-1)}\right)$$

5: **else**
6: Set $n = n+1$ and GOTO 2
7: **end if**

Lemma 3.5. *Assume (I), (II), (III). Then Algorithm 12 outputs a roll of a* $\boldsymbol{p}$*-die and the probability that it needs* $N > n$ *iterations equals*

$$\mathbb{P}(N > n) = 1 - \mathbb{E}\left[\left\|\boldsymbol{L}^{(n)}\right\|_1\right].$$

Proof. Let N be the first iteration where the algorithm stops, then by conditions (I) and (II) it follows:

$$\begin{aligned}
\mathbb{P}(N > n) &= \mathbb{E}\left[\mathbb{I}\left\{U > \left\|\boldsymbol{L}^{(n)}\right\|_1\right\}\right] \\
&= 1 - \mathbb{E}\left[\mathbb{I}\left\{U \leq \left\|\boldsymbol{L}^{(n)}\right\|_1\right\}\right] \\
&= 1 - \mathbb{E}\left[\mathbb{E}\left[\mathbb{I}\left\{U \leq \left\|\boldsymbol{L}^{(n)}\right\|_1\right\}\middle|\boldsymbol{L}^{(n)}\right]\right] \\
&= 1 - \mathbb{E}\left[\left\|\boldsymbol{L}^{(n)}\right\|_1\right].
\end{aligned}$$

Let X be the output of the algorithm and in the following computation note

that $\mathbb{E}\left[\mathbb{I}(\{X=i\}\cap\{N=n\})|\mathcal{F}_{0:\infty}\right]$ is determined in step 4.

$$\begin{aligned}
\mathbb{P}(X=i) &= \mathbb{E}[\mathbb{I}\{X=i\}] \\
&= \mathbb{E}\left[\sum_{n=1}^{\infty}\mathbb{I}\big(\{X=i\}\cap\{N=n\}\big)\right] \\
&= \mathbb{E}\left[\sum_{n=1}^{\infty}\mathbb{E}\left[\mathbb{I}\big(\{X=i\}\cap\{N=n\}\big)\Big|\mathcal{F}_{0:\infty}\right]\right] \\
&= \mathbb{E}\left[\sum_{n=1}^{\infty}\left(L_i^{(n)}-L_i^{(n-1)}\right)\right] = \mathbb{E}\left[\lim_{n\to\infty}L_i^{(n)}\right] \\
&= \lim_{n\to\infty}\mathbb{E}\left[L_i^{(n)}\right] = p_i,
\end{aligned}$$

where the monotone convergence theorem guarantees that it is possible to interchange limit and expectation. □

Via reverse time martingales

We weaken condition (II) and just ask for $\boldsymbol{L}^{(n)}$ to be a reverse time supermartingale:

(IV) $\mathbb{E}\left[\boldsymbol{L}^{(n-1)}|\mathcal{F}_{n:\infty}\right] = \mathbb{E}\left[\boldsymbol{L}^{(n-1)}|\mathcal{F}_{n}\right] \leq \boldsymbol{L^{(n)}}$ a.s.

Then, the following algorithm samples from a $\boldsymbol{p}$-die, assuming that we have a way to compute $\mathbb{E}\left[\boldsymbol{L}^{(n-1)}|\mathcal{F}_{n}\right]$.

Algorithm 13 Via Reverse Time Martingales

1: Set $\boldsymbol{L}^{(0)} := \widetilde{\boldsymbol{L}}^{(0)} := \mathbf{0}$, $C^{(0)} := 1$, $n := 1$, and simulate $U \sim U(0,1)$
2: Obtain $\boldsymbol{L}^{(n)}$ given $\mathcal{F}_{0:(n-1)}$
3: Compute $\widehat{\boldsymbol{L}}^{(n)} := \mathbb{E}\left[\boldsymbol{L}^{(n-1)} | \mathcal{F}_n\right]$ and set

$$\widetilde{\boldsymbol{L}}^{(n)} = \widetilde{\boldsymbol{L}}^{(n-1)} + C^{(n-1)} \left(\frac{\boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}}{1 - \left\| \widehat{\boldsymbol{L}}^{(n)} \right\|_1} \right)$$

4: **if** $U < \left\| \widetilde{\boldsymbol{L}}^{(n)} \right\|_1$ **then**
5: **return** $i \in \{0, \dots, m\}$ such that

$$\left\| \widetilde{\boldsymbol{L}}^{(n-1)} \right\|_1 + \sum_{j=0}^{i-1} \left(\widetilde{L}_j^{(n)} - \widetilde{L}_j^{(n-1)} \right) \leq U < \left\| \widetilde{\boldsymbol{L}}^{(n-1)} \right\|_1 + \sum_{j=0}^{i} \left(\widetilde{L}_j^{(n)} - \widetilde{L}_j^{(n-1)} \right)$$

6: **else**
7: Set $C^{(n)} = C^{(n-1)} \left(\frac{1 - \left\| \boldsymbol{L}^{(n)} \right\|_1}{1 - \left\| \widehat{\boldsymbol{L}}^{(n)} \right\|_1} \right)$, $n = n + 1$, and GOTO 2
8: **end if**

Theorem 3.6. *Assume (I), (III), (IV). Then Algorithm 13 outputs a roll of a $\boldsymbol{p}$-die and the probability that it needs $N > n$ iterations equals*

$$\mathbb{P}(N > n) = 1 - \mathbb{E}\left[\left\| \boldsymbol{L}^{(n)} \right\|_1 \right].$$

Proof. We prove the theorem by noticing that $\widetilde{\boldsymbol{L}}^{(n)}$ is a sequence satisfying (I), (II), (III), and conclude from Lemma 3.5. By assumption, $\widehat{\boldsymbol{L}}^{(n)} \leq \boldsymbol{L}^{(n)}$ implying $C^{(n)} \geq 0$ and

$$\widetilde{\boldsymbol{L}}^{(n)} - \widetilde{\boldsymbol{L}}^{(n-1)} = C^{(n-1)} \left(\frac{\boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}}{1 - \left\| \widehat{\boldsymbol{L}}^{(n)} \right\|_1} \right) \geq 0,$$

so that $\widetilde{\boldsymbol{L}}^{(n)} \geq \mathbf{0}$ for all n, and also condition (II) is satisfied. By induction, we prove

that $C^{(n)} = 1 - \left\|\widetilde{\boldsymbol{L}}^{(n)}\right\|_1$ by noticing:

$$\begin{aligned}\left\|\widetilde{\boldsymbol{L}}^{(n)}\right\|_1 &= \left\|\widetilde{\boldsymbol{L}}^{(n-1)}\right\|_1 + C^{(n-1)}\left(\frac{\left\|\boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}\right\|_1}{1 - \left\|\widehat{\boldsymbol{L}}^{(n)}\right\|_1}\right)\\ &= 1 - C^{(n-1)}\left(1 - \frac{\left\|\boldsymbol{L}^{(n)}\right\|_1 - \left\|\widehat{\boldsymbol{L}}^{(n)}\right\|_1}{1 - \left\|\widehat{\boldsymbol{L}}^{(n)}\right\|_1}\right) = 1 - C^{(n)},\end{aligned}$$

where the equality $\left\|\boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}\right\|_1 = \left\|\boldsymbol{L}^{(n)}\right\|_1 - \left\|\widehat{\boldsymbol{L}}^{(n)}\right\|_1$ holds by assumption (IV). Therefore, because $0 \leq \left\|\widehat{\boldsymbol{L}}^{(n)}\right\|_1 \leq \left\|\boldsymbol{L}^{(n)}\right\|_1 \leq 1$, the sequence $C^{(n)}$ is non-increasing and non-negative, hence $0 \leq \left\|\widetilde{\boldsymbol{L}}^{(n)}\right\|_1 \leq 1$, and (I) is satisfied.

We now prove by induction on n that $\mathbb{E}\left[\widetilde{\boldsymbol{L}}^{(n)}\right] = \mathbb{E}\left[\boldsymbol{L}^{(n)}\right]$, thus demonstrating that also condition (III) holds. Using $\widehat{\boldsymbol{L}}^{(1)} = \mathbf{0}$, note that $\mathbb{E}\left[\widetilde{\boldsymbol{L}}^{(1)}\right] = \mathbb{E}\left[\boldsymbol{L}^{(1)}\right]$ and compute:

$$\begin{aligned}\mathbb{E}\left[\widetilde{\boldsymbol{L}}^{(n)}\right] - \mathbb{E}\left[\boldsymbol{L}^{(n-1)}\right] &= \mathbb{E}\left[\mathbb{E}\left[C^{(n-1)}\left(\frac{\boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}}{1 - \left\|\widehat{\boldsymbol{L}}^{(n)}\right\|_1}\right)|\mathcal{F}_{2:\infty}\right]\right]\\ &= \mathbb{E}\left[\mathbb{E}\left[\frac{\boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}}{1 - \left\|\widehat{\boldsymbol{L}}^{(n)}\right\|_1}\cdot\frac{1 - \left\|\boldsymbol{L}^{(n-1)}\right\|_1}{1 - \left\|\widehat{\boldsymbol{L}}^{(n-1)}\right\|_1}\cdot\ldots\cdot\frac{1 - \left\|\boldsymbol{L}^{(1)}\right\|_1}{1 - \left\|\widehat{\boldsymbol{L}}^{(1)}\right\|_1}|\mathcal{F}_{2:\infty}\right]\right]\\ &= \mathbb{E}\left[\frac{\boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}}{1 - \left\|\widehat{\boldsymbol{L}}^{(n)}\right\|_1}\cdot\frac{1 - \left\|\boldsymbol{L}^{(n-1)}\right\|_1}{1 - \left\|\widehat{\boldsymbol{L}}^{(n-1)}\right\|_1}\cdot\ldots\cdot\frac{1 - \left\|\boldsymbol{L}^{(2)}\right\|_1}{1 - \left\|\widehat{\boldsymbol{L}}^{(2)}\right\|_1}\mathbb{E}\left[\left(1 - \left\|\boldsymbol{L}^{(1)}\right\|_1\right)|\mathcal{F}_{2:\infty}\right]\right]\\ &= \mathbb{E}\left[\frac{\boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}}{1 - \left\|\widehat{\boldsymbol{L}}^{(n)}\right\|_1}\cdot\frac{1 - \left\|\boldsymbol{L}^{(n-1)}\right\|_1}{1 - \left\|\widehat{\boldsymbol{L}}^{(n-1)}\right\|_1}\cdot\ldots\cdot\frac{1 - \left\|\boldsymbol{L}^{(2)}\right\|_1}{1 - \left\|\widehat{\boldsymbol{L}}^{(2)}\right\|_1}(1 - \left\|\widehat{\boldsymbol{L}}^{(2)}\right\|_1)\right]\\ &= \mathbb{E}\left[\frac{\boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}}{1 - \left\|\widehat{\boldsymbol{L}}^{(n)}\right\|_1}\cdot\frac{1 - \left\|\boldsymbol{L}^{(n-1)}\right\|_1}{1 - \left\|\widehat{\boldsymbol{L}}^{(n-1)}\right\|_1}\cdot\ldots\cdot\frac{1 - \left\|\boldsymbol{L}^{(3)}\right\|_1}{1 - \left\|\widehat{\boldsymbol{L}}^{(3)}\right\|_1}\mathbb{E}\left[\left(1 - \left\|\boldsymbol{L}^{(2)}\right\|_1\right)|\mathcal{F}_{3:\infty}\right]\right]\\ &= \ldots = \mathbb{E}\left[\boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}\right],\end{aligned}$$

so that

$$\mathbb{E}\left[\widetilde{\boldsymbol{L}}^{(n)}\right] = \mathbb{E}\left[\mathbb{E}[\boldsymbol{L}^{(n-1)} + \boldsymbol{L}^{(n)} - \widehat{\boldsymbol{L}}^{(n)}|\mathcal{F}_{n:\infty}]\right] = \mathbb{E}\left[\boldsymbol{L}^{(n)}\right].$$

□

3.3 Extending the Bernoulli Factory to a Dice Enterprise

We now extend the classic Bernoulli Factory setting to a multivariate one, renamed *Dice Enterprise*. We give a formal definition of the problem, adapting the one for Bernoulli Factories presented in Section 1.1.

Definition 3.7 (Dice Enterprise)**.** Given an unknown vector $\boldsymbol{p} \in S \subset \Delta^m$ and a known function $f(\boldsymbol{p}) : S \subset \Delta^m \to \Delta^v$, let $\mathcal{A}$ be a computable function that takes as input a number $u \in [0,1]$ and a sequence of values in $\{0,\dots,m\}$, and returns an output in $\{0,\dots,v\}$. For any sequence $\left(X^{(n)}\right)_{n\in\mathbb{N}^+} \overset{iid}{\sim} \boldsymbol{p}$ and $U \sim \text{Unif}(0,1)$, let

$$T := \inf\left\{n : \mathcal{A}\left(U, X^{(1)}, X^{(2)} \dots\right) = \mathcal{A}\left(U, X^{(1)}, X^{(2)}, \dots, X^{(n)}\right)\right\}.$$

Assume the following holds:

- T is a stopping time with respect to the natural filtration and is almost surely finite.
- $\mathcal{A}(U, X^{(1)}, X^{(2)}, \dots) \sim f(\boldsymbol{p})$.

Then $\mathcal{A}$ is a *Dice Enterprise* for f.

If a Dice Enterprise for $f(\boldsymbol{p})$ exists, we say that there exists an algorithm that tosses an $f(\boldsymbol{p})$-die. As in the Bernoulli Factory case, the introduction of the source of randomness $U \sim \text{Unif}(0,1)$ in the above definition eases the analysis and notation, but is not necessary from a theoretical standpoint. Indeed, as we do not allow for $\boldsymbol{p} \in \bar{\Delta}^m$, we can make use of the $\boldsymbol{p}$-die to generate a uniform random variable to any arbitrary precision, as shown in Section 1.4.

Crucially, in the following we can just consider Dice Enterprises for functions $f(\boldsymbol{p}) : S \subset \Delta^m \to (0,1)$, that is Dice Enterprises that take dice rolls as input and return coin tosses. To see why this is the case, let $f(\boldsymbol{p}) : S \subset \Delta^m \to \Delta^v$ and consider constructing $v+1$ different Dice Enterprises targeting $f_0(\boldsymbol{p}), \dots, f_v(\boldsymbol{p})$ respectively. Therefore, we have a way to toss independently an $f_0(\boldsymbol{p})$-coin, $f_1(\boldsymbol{p})$-coin, etc. and can resort to already available algorithms, such as the Bernoulli race algorithm proposed by Dughmi et al. [12], to construct an $f(\boldsymbol{p})$-die as desired. We make this formal in the following lemma.

Lemma 3.8. *Given $f : S \subset \Delta^m \to \Delta^v$, a Dice Enterprise for $f(\boldsymbol{p})$ exists if and only if there exist algorithms that toss $f_0(\boldsymbol{p}), \dots, f_v(\boldsymbol{p})$-coins.*

Proof. $\Rightarrow$ Roll the $f(\boldsymbol{p})$-die with the given algorithm. If the i^{th} face is rolled output heads, otherwise output tails. Then, this procedure returns heads with probability $f_i(\boldsymbol{p})$.
$\Leftarrow$ Use rejection sampling. Propose X distributed uniformly on $\{0, \dots, v\}$ (cf. Section 1.4). Toss the $f_X(\boldsymbol{p})$-coin and accept X if the coin is heads, otherwise restart. □

Throughout the remaining of this section, we extend the results of Keane and O'Brien [30], Nacu and Peres [43] and discuss: (i) construction of Dice Enterprise for functions $f(\boldsymbol{p})$ that take constant values, (ii) construction of Dice Enterprise for generic functions $f(\boldsymbol{p})$ via bounding polynomials, (iii) algorithmic implementation for twice continuous differentiable functions, and (iv) sufficient and necessary conditions on $f(\boldsymbol{p})$ for a Dice Enterprise to exist.

3.3.1 Constructing a Dice Enterprise

Assume a Dice Enterprise for a given function $f(\boldsymbol{p}) : \Delta^m \to \Delta^v$ exists. This implies that we can use a stream of rolls – which we can view as a string made of characters belonging to $\{0, \dots, m\}$ – to determine a roll of the $f(\boldsymbol{p})$-die. In other words, the algorithm scans bit by bit the given string and at each stage decides whether it can stop, outputting a result, or needs to continue and read the next character. Therefore, it is natural to define sets $A_i^{(n)}$ consisting of n-long strings for which the algorithm terminates and outputs i. One can immediately see that some constraints arise: for instance if $m = 2$ and the string `021` belongs to $A_1^{(3)}$, then the strings `0210`, `0211` and `0212` must necessarily belong to $A_1^{(4)}$. Therefore, a Dice Enterprise *algorithm* automatically defines such *sets*, although in practice it may be hard to compute them.

Each element of $A_i^{(n)}$ has a specific probability of being observed, which is a polynomial of order n in $\boldsymbol{p}$. Therefore, it is hopefully now clear that if a Dice Enterprise exists, then it uniquely defines such *polynomials*. Perhaps unsurprisingly, one can prove that the opposite also holds: given polynomials satisfying specific conditions (corresponding to the constraints aforementioned), they uniquely define a Dice Enterprise type algorithm. We make this formal in Theorem 3.9, which is in spirit similar to those of Nacu and Peres [43] (Proposition 3) and Łatuszyński et al. [32] (Proposition 3.1), here extended to the multivariate setting, and make use of the framework we developed in Section 3.2.

Theorem 3.9. *Let $f : S \subset \Delta^m \to \Delta^v$ denoted as $f(\boldsymbol{p}) = (f_0(\boldsymbol{p}), \dots, f_v(\boldsymbol{p}))$. A Dice Enterprise for f exists if and only if for all $n \geq 0$ and $i \in \{0, \dots, v\}$ there exist polynomials*

$$f_i^{(n)}(\boldsymbol{p}) = \sum_{\boldsymbol{k} \in \Lambda_n^m} \binom{n}{\boldsymbol{k}} l_i^{(n)}(\boldsymbol{k}) \boldsymbol{p}^{\boldsymbol{k}},$$

where

1. *$l_i^{(n)}(\boldsymbol{k}) \geq 0$, and $\sum_{i=0}^v l_i^{(n)}(\boldsymbol{k}) \leq 1$, $\forall \boldsymbol{k} \in \Lambda_n^m$;*
2. *$\lim_{n\to\infty} f_i^{(n)}(\boldsymbol{p}) = f_i(\boldsymbol{p})$ for all $\boldsymbol{p} \in S, i \in \{0, \dots, v\}$;*
3. *For all $s < n$, it holds*

$$l_i^{(n)}(\boldsymbol{k}) \geq \sum_{\boldsymbol{k}' \in \Lambda_s^m, \boldsymbol{k} \geq \boldsymbol{k}'} \frac{\binom{n-s}{\boldsymbol{k}-\boldsymbol{k}'}\binom{s}{\boldsymbol{k}'}}{\binom{n}{\boldsymbol{k}}} l_i^{(s)}(\boldsymbol{k}').$$

Moreover, let N be the number of rolls of the $\boldsymbol{p}$-die required for the algorithm to stop. Then, $\mathbb{P}_{\boldsymbol{p}}(N > n) = 1 - \sum_{i=0}^v f_i^{(n)}(\boldsymbol{p})$.

Proof. Polynomials ⇒ Algorithm

Let $X^{(1)}, X^{(2)}, \dots$ be a sequence of independent rolls of the $\boldsymbol{p}$-die and let $\boldsymbol{W}^{(n)} = \left(W_0^{(n)}, \dots, W_m^{(n)}\right)$ such that $W_i^{(n)}$ is the number of times the i^{th} face has been rolled, that is $W_i^{(n)} = \sum_{t=1}^n \mathbb{I}\{X^{(t)} = i\}$. Let $\boldsymbol{L}^{(n)}$ be defined by setting $L_i^{(n)} = l_i^{(n)}\left(\boldsymbol{W}^{(n)}\right)$. We show that conditions (I), (III), (IV) hold, so that the assumptions of Theorem 3.6 are met and Algorithm 13 outputs a sample from an $f(\boldsymbol{p})$-die.

Condition (I) and the convergence of expectation in (III) follow easily from (1), (2). For (IV) and for the monotonicity in (III), define the σ-algebra $\mathcal{G}_n = \sigma\left(\boldsymbol{W}^{(n)}\right)$ and note that for $s < n$ the distribution of $\boldsymbol{W}^{(n-s)}$ given $\boldsymbol{W}^{(n)}$ is multivariate hypergeometric. Hence, using assumption (3) we can compute:

$$\begin{aligned}
\mathbb{E}\left[L_i^{(n-s)} | \mathcal{G}_n\right] &= \mathbb{E}\left[l_i^{(n-s)}\left(\boldsymbol{W}^{(n-s)}\right) | \mathcal{G}_n\right] \\
&= \mathbb{E}\left[l_i^{(n-s)}\left(\boldsymbol{W}^{(n-s)}\right) | \boldsymbol{W}^{(n)}\right] \\
&= \sum_{\boldsymbol{k}' \in \Lambda_{n-s}^m :\, \boldsymbol{k}' \leq \boldsymbol{W}^{(n)}} \frac{\binom{s}{\boldsymbol{W}^{(n)}-\boldsymbol{k}'}\binom{n-s}{\boldsymbol{k}'}}{\binom{n}{\boldsymbol{W}^{(n)}}} l_i^{(n-s)}(\boldsymbol{k}') \\
&\leq l_i^{(n)}\left(\boldsymbol{W}^{(n)}\right) = L_i^{(n)}.
\end{aligned}$$

This yields (IV) and after taking expectations of both sides, also implies monotonicity of convergence in (III).

Let N be the number of iterations required for the algorithm to terminate; by Theorem 3.6 it follows:

$$\mathbb{P}(N > n) = 1 - \mathbb{E}\left[\left\|\boldsymbol{L}^{(n)}\right\|_1\right] = 1 - \sum_{i=0}^{v} \mathbb{E}\left[L_i^{(n)}\right] = 1 - \sum_{i=0}^{v} f_i^{(n)}(\boldsymbol{p}),$$

and since by condition (2) it follows $\lim_{n\to\infty} \sum_{i=0}^{v} f_i^{(n)}(\boldsymbol{p}) = 1$, the algorithm terminates almost surely as desired.

Algorithm ⇒ Polynomials

Let $X^{(1)}, X^{(2)}, \ldots, X^{(n)}$ be n rolls of the $\boldsymbol{p}$-die. For $i \in \{0, \ldots, v\}$, define $A_i^{(n)}$ as the set of n-long inputs where the algorithm terminates and outputs i. It follows:

$$\mathbb{P}\left(A_i^{(n)}\right) \leq \mathbb{P}(\text{algorithm outputs } i) = f_i(\boldsymbol{p}) \tag{3.1}$$

Partition the set $A_i^{(n)}$ according to how many times each face of the $\boldsymbol{p}$-die has been rolled:

$$A_i^{(n)} = \bigcup_{\boldsymbol{k}\in\Lambda_n^m} A_{i,\boldsymbol{k}}^{(n)},$$

and notice that the probability of observing each input in $A_{i,\boldsymbol{k}}^{(n)}$ is given by $\boldsymbol{p}^{\boldsymbol{k}}$. Define:

$$l_i^{(n)}(\boldsymbol{k}) = \left|A_{i,\boldsymbol{k}}^{(n)}\right| / \binom{n}{\boldsymbol{k}}.$$

It then follows:

$$\mathbb{P}_{\boldsymbol{p}}\left(A_i^{(n)}\right) = \mathbb{P}_{\boldsymbol{p}}\left(\bigcup_{\boldsymbol{k}\in\Lambda_n^m} A_{i,\boldsymbol{k}}^{(n)}\right) = \sum_{\boldsymbol{k}\in\Lambda_n^m} \left|A_{i,\boldsymbol{k}}^{(n)}\right| \boldsymbol{p}^{\boldsymbol{k}} = \sum_{\boldsymbol{k}\in\Lambda_n^m} l_i^{(n)}(\boldsymbol{k}) \binom{n}{\boldsymbol{k}} \boldsymbol{p}^{\boldsymbol{k}}.$$

Define $f_i^{(n)}(\boldsymbol{p}) = \sum_{\boldsymbol{k}\in\Lambda_n^m} \binom{n}{\boldsymbol{k}} l_i^{(n)}(\boldsymbol{k}) \boldsymbol{p}^{\boldsymbol{k}}$. We now prove that polynomials $f_i^{(n)}(\boldsymbol{p})$ satisfy conditions (1)-(3):

(1) Clearly, by definition, $l_i^{(n)}(\boldsymbol{k}) \geq 0$. For a fixed $\boldsymbol{k} \in \Lambda_n^m$, let $\Omega_{\boldsymbol{k}}^{(n)}$ be the set of all sequences of length n such that there are k_0 zeros, k_1 ones, …, k_m m's. Then $\left|\Omega_{\boldsymbol{k}}^{(n)}\right| = \binom{n}{\boldsymbol{k}}$. Note that $\cup_{i=0}^{v} A_{i,\boldsymbol{k}}^{(n)}$ is the set of n-long inputs such that the algorithm terminates and there are k_0 zeros, k_1 ones, …, k_m m's. Therefore

$\cup_{i=0}^{v} A_{i,\boldsymbol{k}}^{(n)} \subset \Omega_{\boldsymbol{k}}^{(n)}$. We conclude by noticing:

$$\sum_{i=0}^{v} l_i^{(n)}(\boldsymbol{k}) = \left|\cup_{i=0}^{v} A_{i,\boldsymbol{k}}^{(n)}\right| / \binom{n}{\boldsymbol{k}} \leq \left|\Omega_{\boldsymbol{k}}^{(n)}\right| / \binom{n}{\boldsymbol{k}} = 1.$$

(2) By definition, $f_i^{(n)}(\boldsymbol{p}) \leq f_i^{(n+1)}(\boldsymbol{p}) \leq f_i(\boldsymbol{p})$. By contradiction, assume (2) does not hold, so that it must be $\lim_{n\to\infty} f_i^{(n)}(\boldsymbol{p}) < f_i(\boldsymbol{p})$. This would imply

$$\lim_{n\to\infty} \sum_{i=0}^{v} f_i^{(n)}(\boldsymbol{p}) = \lim_{n\to\infty} \sum_{i=0}^{v} \mathbb{P}(A_i^{(n)}) < 1,$$

and the algorithm would not terminate almost surely.

(3) Notice that if a sequence $\left(X^{(1)}, \ldots, X^{(s)}\right)$ is in $A_i^{(s)}$ then for every $n > s$ any sequence starting with the same $\left(X^{(1)}, \ldots, X^{(s)}\right)$ must be in $A_i^{(n)}$ as well. More formally, we shall denote by $T_n\left(A_i^{(s)}\right)$ the set of sequences of length n that have as first s input a sequence in $A_i^{(s)}$. Then, it follows $T_n\left(A_i^{(s)}\right) \subset A_i^{(n)}$. As before, we can partition the set $T_n\left(A_i^{(s)}\right)$ according to how many times each face has been rolled and write $T_n\left(A_i^{(s)}\right) = \bigcup_{\boldsymbol{k}\in\Lambda_n^m} T_{n,\boldsymbol{k}}\left(A_i^{(s)}\right)$. Notice that also $T_{n,\boldsymbol{k}}\left(A_i^{(s)}\right) \subset A_{i,\boldsymbol{k}}^{(n)}$. The cardinality of the set $T_{n,\boldsymbol{k}}\left(A_i^{(s)}\right)$ is given by:

$$\left|T_{n,\boldsymbol{k}}\left(A_i^{(s)}\right)\right| = \sum_{\boldsymbol{k'}\in\Lambda_s^m, \boldsymbol{k}\geq\boldsymbol{k'}} \binom{n-s}{\boldsymbol{k}-\boldsymbol{k'}} \left|A_{i,\boldsymbol{k'}}^{(s)}\right|.$$

Since $T_{n,\boldsymbol{k}}\left(A_i^{(s)}\right) \subset A_{i,\boldsymbol{k}}^{(n)}$, it follows that $\left|A_{i,\boldsymbol{k}}^{(n)}\right| \geq \left|T_{n,\boldsymbol{k}}\left(A_i^{(s)}\right)\right|$ and thus

$$\binom{n}{\boldsymbol{k}} l_i^{(n)}(\boldsymbol{k}) \geq \sum_{\boldsymbol{k'}\in\Lambda_s^m, \boldsymbol{k}\geq\boldsymbol{k'}} \binom{n-s}{\boldsymbol{k}-\boldsymbol{k'}} \binom{s}{\boldsymbol{k'}} l_i^{(s)}(\boldsymbol{k'}),$$

as desired.

□

Remark 3.10. Given polynomials satisfying the hypothesis of Theorem 3.9, one can explicitly implement the Dice Enterprise algorithm by applying Algorithm 13 (cf. theorem's proof). Moreover, the theorem still holds if the polynomials $f_i^{(n)}(\boldsymbol{p})$ are defined for an increasing subsequence $\{n_i\}_{i\in\mathbb{N}}$. This requires a small modification of

line 7 of Algorithm 13, where instead of increasing n by one we consider the next value in the sequence $\{n_i\}_{i\in\mathbb{N}}$.

The form of the polynomials of Theorem 3.9 is closely related to that of multivariate Bernstein's polynomials [11]. Recall that if $f : \Delta^m \to \mathbb{R}$ is a continuous function, its Bernstein's polynomial approximation of order n is defined as:

$$\sum_{\boldsymbol{k}\in\Lambda_n^m} \binom{n}{\boldsymbol{k}} f\left(\frac{k_0}{n},\dots,\frac{k_m}{n}\right)\boldsymbol{p}^{\boldsymbol{k}},$$

and converges uniformly to f as $n\to\infty$. Therefore one can aim to carefully tweak Bernstein polynomials to find suitable polynomials as in Theorem 3.9. We show one example of that in the following proposition, where we multiply Bernstein's polynomials by suitable constants so to derive an algorithm applicable to any twice differentiable function. The idea is in spirit similar to that of Nacu and Peres [43] (Proposition 10), however it relaxes some of the hypothesis as not only it is applicable to the new multivariate setting, but it also does not require for the function f to be bounded in $[\epsilon, 1-\epsilon], \epsilon > 0$.

Proposition 3.11. *Let $f(\boldsymbol{p}) : \bar{\Delta}^m \to [0,1]$ be a twice continuously differentiable function and let H be the Hessian matrix. If $\|H\|_2 \le C$, then there exists an algorithm to toss an $f(\boldsymbol{p})$-coin. Moreover, let N be the number of rolls required for the algorithm to terminate, then:*

$$\mathbb{P}(N > n) = 1 - \frac{\Gamma(n-\sqrt{-C})\Gamma(n+\sqrt{-C})}{\Gamma(n)^2}$$

Proof. Let

$$\epsilon^{(n)} := \frac{\Gamma(n-\sqrt{-C})\Gamma(n+\sqrt{-C})}{\Gamma(n)^2},$$

and let $X^{(1)}, X^{(2)}, \dots$ be a sequence of independent rolls of the $\boldsymbol{p}$-die. Consider the vector $\boldsymbol{W}^{(n)}$ which counts how many times each face has been rolled, i.e. $W_i^{(n)} = \sum_{t=1}^n \mathbb{I}\{X^{(t)} = i\}$. Then, define the sequence $\boldsymbol{L}^{(n)} = \left(L_0^{(n)}, L_1^{(n)}\right)$ given by:

$$L_0^{(n)} = \epsilon^{(n)}\left(1 - f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right)\right), \qquad L_1^{(n)} = \epsilon^{(n)} f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right).$$

We now prove that $\boldsymbol{L}^{(n)}$ satisfies conditions (I), (III), (IV), so that Algorithm 13 can be used. By Theorem 3.6 the tail bound on the number N of rolls required is as desired.

Notice that $\{\epsilon^{(n)}\}_{n\in\mathbb{N}}$ is an increasing sequence tending to 1, therefore condition (I) is trivially satisfied. Moreover:

$$\mathbb{E}\left[L_1^{(n)}\right] = \epsilon^{(n)} \sum_{\boldsymbol{k}\in\Lambda_n^m} \binom{n}{\boldsymbol{k}} \boldsymbol{p}^{\boldsymbol{k}} f\left(\frac{\boldsymbol{k}}{n}\right),$$

and as discussed in Section 3.3.1, it converges uniformly to $f(\boldsymbol{p})$, since the expected value is proportional to the multivariate Bernstein's polynomial and $\epsilon^{(n)} \to 1$. We conclude the same for $\mathbb{E}\left[L_0^{(n)}\right]$. We are now left to prove that also condition (IV) holds. Notice that:

$$\begin{aligned}
\mathbb{E}\left[L_1^{(n-1)}|\mathcal{F}_n\right] = \mathbb{E}\left[L_1^{(n-1)}|\boldsymbol{W}^{(n)}\right] &= \sum_{\boldsymbol{k}'\in\Lambda_{n-1}^m, \boldsymbol{k}'\leq \boldsymbol{W}^{(n)}} \frac{\binom{1}{\boldsymbol{W}^{(n)}-\boldsymbol{k}'}\binom{n-1}{\boldsymbol{k}'}}{\binom{n}{\boldsymbol{W}^{(n)}}} \epsilon^{(n-1)} f\left(\frac{\boldsymbol{k}'}{n-1}\right) \\
&= \epsilon^{(n-1)} \sum_{j=0}^{m} \frac{\binom{1}{\boldsymbol{e}_j}\binom{n-1}{\boldsymbol{W}^{(n)}-\boldsymbol{e}_j}}{\binom{n}{\boldsymbol{W}^{(n)}}} f\left(\frac{\boldsymbol{W}^{(n)}-\boldsymbol{e}_j}{n-1}\right) \\
&= \epsilon^{(n-1)} \sum_{j=0}^{m} \frac{(W_0^{(n)})!\ldots(W_j^{(n)})!\ldots(W_m^{(n)})!}{(W_0^{(n)})!\ldots(W_j^{(n)}-1)!\ldots(W_m^{(n)})!} \frac{(n-1)!}{n!} f\left(\frac{\boldsymbol{W}^{(n)}-\boldsymbol{e}_j}{n-1}\right) \\
&= \epsilon^{(n-1)} \sum_{j=0}^{m} \frac{W_j^{(n)}}{n} f\left(\frac{\boldsymbol{W}^{(n)}-\boldsymbol{e}_j}{n-1}\right).
\end{aligned}$$

Therefore, condition (IV) can be rewritten as:

$$\epsilon^{(n-1)} \sum_{j=0}^{m} \frac{W_j^{(n)}}{n} f\left(\frac{\boldsymbol{W}^{(n)}-\boldsymbol{e}_j}{n-1}\right) \leq \epsilon^{(n)} f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right). \tag{3.2}$$

Since f is twice differentiable and $\|H\|_2 \leq C$, a truncated Taylor expansion gives:

$$\begin{aligned}
f\left(\frac{\boldsymbol{W}^{(n)}-\boldsymbol{e}_j}{n-1}\right) \leq f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right) &+ \nabla f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right)^T \left(\frac{\boldsymbol{W}^{(n)}-n\boldsymbol{e}_j}{n(n-1)}\right) \\
&+ \frac{C}{2}\left(\frac{\boldsymbol{W}^{(n)}-n\boldsymbol{e}_j}{n(n-1)}\right)^T \left(\frac{\boldsymbol{W}^{(n)}-n\boldsymbol{e}_j}{n(n-1)}\right)
\end{aligned}$$

Notice the following:

- Since $\left\|\boldsymbol{W}^{(n)}\right\|_1 = n$:

$$\sum_{j=0}^{m} \frac{W_j^{(n)}}{n} f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right) = f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right)$$

-

$$\sum_{j=0}^{m} \frac{W_j^{(n)}}{n} \nabla f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right)^T \left(\frac{\boldsymbol{W}^{(n)} - n\boldsymbol{e}_j}{n(n-1)}\right) = \mathbf{0}$$

  To see why, denote $\boldsymbol{Z} := \sum_{j=0}^{m} W_j^{(n)} \left(\boldsymbol{W}^{(n)} - n\boldsymbol{e}_j\right)$ and notice:

$$Z_j = W_j(W_j - n) + W_j \sum_{h \neq j} W_h = W_j(W_j - n) + W_j(n - W_j) = 0$$

- Since $\left\|\boldsymbol{W}^{(n)}\right\|_1 = n$, we obtain the following bound:

$$\begin{aligned}
&\frac{C}{2} \sum_{j=0}^{m} \frac{W_j^{(n)}}{n} \left(\frac{\boldsymbol{W}^{(n)} - n\boldsymbol{e}_j}{n(n-1)}\right)^T \left(\frac{\boldsymbol{W}^{(n)} - n\boldsymbol{e}_j}{n(n-1)}\right) \\
&\quad = \frac{C}{2} \sum_{j=0}^{m} \frac{W_j^{(n)}}{n^3(n-1)^2} \left[\left(W_j^{(n)} - n\right)^2 + \sum_{h \neq j} \left(W_h^{(n)}\right)^2\right] \\
&\quad \leq \frac{C}{2} \sum_{j=0}^{m} \frac{W_j^{(n)}}{n^3(n-1)^2} \left[n^2 + \sum_{h=0}^{m} \left(W_h^{(n)}\right)^2\right] \\
&\quad \leq \frac{C}{2} \sum_{j=0}^{m} \frac{W_j^{(n)}}{n^3(n-1)^2} \left[n^2 + \left(\sum_{h=0}^{m} W_h^{(n)}\right)^2\right] \\
&\quad = \frac{C}{2} \sum_{j=0}^{m} \frac{W_j^{(n)}}{n^3(n-1)^2} \left[2n^2\right] \leq C \sum_{j=0}^{m} \frac{W_j^{(n)}}{n(n-1)^2} \leq \frac{C}{(n-1)^2}
\end{aligned}$$

Putting all together, the LHS of eq. (3.2) can be bounded as:

$$\epsilon^{(n-1)} \sum_{j=0}^{m} \frac{W_j^{(n)}}{n} f\left(\frac{\boldsymbol{W}^{(n)} - \boldsymbol{e}_j}{n-1}\right) \leq \epsilon^{(n-1)} \left[f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right) + \frac{C}{(n-1)^2}\right],$$

so that eq. (3.2) is surely satisfied if we impose:

$$\epsilon^{(n)} f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right) \geq \epsilon^{(n-1)}\left[f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right) + \frac{C}{(n-1)^2}\right],$$

and since $f(\boldsymbol{p}) \leq 1$ we can further write:

$$\frac{C}{(n-1)^2} \leq f\left(\frac{\boldsymbol{W}^{(n)}}{n}\right)\left(\frac{\epsilon^{(n)}}{\epsilon^{(n-1)}} - 1\right) \leq \left(\frac{\epsilon^{(n)} - \epsilon^{(n-1)}}{\epsilon^{(n-1)}}\right),$$

and so it is enough to have:

$$\frac{\epsilon^{(n)}}{\epsilon^{(n-1)}} \geq \left(1 + \frac{C}{(n-1)^2}\right).$$

Given how we defined $\epsilon^{(n)}$ it is indeed the case that $\frac{\epsilon^{(n)}}{\epsilon^{(n-1)}} = \left(1 + \frac{C}{(n-1)^2}\right)$. Analogous calculations can be carried out for $\mathbb{E}\left[L_0^{(n)}\right]$, concluding the proof. □

Example 3.12. Consider constructing a Bernoulli Factory for $f(p) = ap$ with $ap \leq 1-\epsilon$. To this end, Flegal and Herbei [15] consider an extended function $\widetilde{f}(p)$ defined as:

$$\widetilde{f}(p) = \begin{cases} ap & \text{if } p \in \left[0, \frac{1-\epsilon}{a}\right), \\ (1-\epsilon) + \delta \int_0^{(ap-1+\epsilon)/\delta} e^{-t^2} dt & \text{if } p \in \left[\frac{1-\epsilon}{a}, 1\right], \end{cases}$$

where $0 < \delta < \epsilon$. The function agrees with $f(p)$ on $[0, 1-\epsilon]$, but it is twice differentiable with $\left|\widetilde{f}''\right| < C$, $C := a^2 \frac{\sqrt{2}}{\delta\sqrt{e}}$. The authors make use of the envelopes proposed by Nacu and Peres [43] for twice differentiable functions and develop an algorithm using an analogous of Algorithm 13 for the Bernoulli Factory case. Let N be the number of tosses required for their proposed algorithm to terminate. The algorithm needs an initial number of tosses n_0 (which depends on the values of a and δ) and it then satisfies:

$$\mathbb{P}(N = n) = \begin{cases} 1 - \frac{C}{2n} & \text{if } n = 2^{n_0}, \\ \frac{C}{2n} & \text{if } n > 2^{n_0} \text{ and } n \text{ is a power of 2}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that the algorithm doubles at each iteration the number of p-coin tosses required. Unfortunately, a simple calculation shows that the expected number of tosses

is infinite. On the other hand, the envelopes proposed in Proposition 3.11 satisfy:

$$\mathbb{P}(N > n) = 1 - \frac{\Gamma(n - \sqrt{-C})\Gamma(n + \sqrt{-C})}{\Gamma(n)^2}$$

and do not require an initial number of tosses, although the final algorithm still exhibits an infinite expected running time.

3.3.2 Existence of a Dice Enterprise

In the previous section we clarified how, given an algorithm, we can derive bounding polynomials and vice-versa. However this does not immediately offer insights into for which type of functions $f(\boldsymbol{p}) : S \subset \Delta^m \to \Delta^v$ a Dice Enterprise may exist. For the Bernoulli Factory case (i.e., $m = v = 1$) necessary and sufficient conditions are known and we have presented them in Theorem 1.2. We prove in the following that the natural generalisation of such conditions to the multivariate settings are also sufficient and necessary.

In light of the discussion of the previous section, one can intuitively derive such conditions: the function needs to be continuous and polynomially bounded away from 0 and 1. Indeed, we showed in Theorem 3.9 that if a Dice Enterprise exists, then $f(\boldsymbol{p})$ can be arbitrarily approximated by polynomials that cannot assume value 0 or 1 in the domain of $\boldsymbol{p}$. Surprisingly, it turns out that these conditions are also sufficient, although proving this result appears more challenging. Our proof is inspired by the one for the Bernoulli Factory case by Wästlund [61], and albeit being quite involved, follows an easy idea:

1. We first show in Lemma 3.13 that for any given open cover of the space S, we can roll the given $\boldsymbol{p}$-die sufficiently many times, so to guess in which set $\boldsymbol{p}$ lies, with an error that can be made arbitrarily small and depends on $\boldsymbol{p}$.

2. In Lemma 3.14 we make use of the previous result and show that we can approximate $f(\boldsymbol{p})$ with a function $g(\boldsymbol{p})$ so that the error $|f(\boldsymbol{p}) - g(\boldsymbol{p})|$ can again be made arbitrarily small for all possible values of $\boldsymbol{p} \in S$. Crucially, we show that we can construct a Dice Enterprise for $g(\boldsymbol{p})$.

3. In Lemma 3.15 we finally show how to obtain a roll of the $f(\boldsymbol{p})$-die via a truncation argument on a series of functions. Such functions are defined via the previous construction.

4. We finally make use of the result of Lemma 3.8 to extend the proofs from the die-to-coin case to the die-to-die case.

Interestingly, the function $f(\boldsymbol{p})$ although being continuous may still present pathological behaviour and for instance oscillate indefinitely as $\boldsymbol{p}$ approaches the border of the simplex. The proof produces an algorithm, that although may be hard to put into practice for a given function $f(\boldsymbol{p})$, can be tracked to define the strings of characters in $\{0,\dots,m\}$ and, therefore, the bounding polynomials for $f(\boldsymbol{p})$.

Lemma 3.13. *Given a $\boldsymbol{p}$-die and an open cover $\{U_i\}_{i\in\mathbb{N}}$ of $S\subset\Delta^m$, there exists an algorithm outputting $X\in\mathbb{N}$ such that for any fixed chosen positive integer k and for all $\boldsymbol{p}\in S$:*

$$\sum_{\alpha:\boldsymbol{p}\notin U_\alpha}\mathbb{P}_{\boldsymbol{p}}(X=\alpha)<\boldsymbol{p}^k. \tag{3.3}$$

Proof. We first consider $S=\Delta^m$ and denote by $\widehat{\boldsymbol{p}}^{(N)}$ an estimate of $\boldsymbol{p}$ obtained via N rolls of the given $\boldsymbol{p}$-die. Such estimate is constructed as follows: consider a sequence of independent rolls $X^{(1)},\dots,X^{(N)}$ of the $\boldsymbol{p}$-die and let $\widehat{\boldsymbol{p}}_i^{(N)}=\frac{1}{N}\sum_{j=1}^N\mathbb{I}\{X^{(j)}=i\}$. Then, the algorithm proceeds as follows:

1. Roll the $\boldsymbol{p}$-die until we have observed at least $(m+1)k+1$ rolls of each face, say on the t^{th} roll. Define the set:

$$E=\left\{\boldsymbol{p}\in\Delta^m:p_i\geq m^{-t},\forall i=0,\dots,m\right\}.$$

2. Since E is a compact set, we can select a finite cover, say $U_{k_1},\dots,U_{k_n}$, from the given cover $\{U_i\}_{i\in\mathbb{N}}$. Moreover, consider an $\epsilon>0$ such that $\boldsymbol{p}^k>\epsilon$ for all $\boldsymbol{p}\in E$.

3. For each U_{k_i}, choose a closed subset $F_i\subset U_{k_i}$ so that $\cup_{i=1}^n F_i$ still covers E.

4. Choose an integer N large enough so that for all $i=1,\dots,n$:

$$\begin{aligned}\mathbb{P}_{\boldsymbol{p}}\left(\widehat{\boldsymbol{p}}^{(N)}\in F_i,\boldsymbol{p}\notin U_{k_i}\right)&<\frac{\epsilon}{n+1},\qquad\text{if }\boldsymbol{p}\in E,\\ \mathbb{P}_{\boldsymbol{p}}\left(\widehat{\boldsymbol{p}}^{(N)}\notin E\right)&<\frac{\epsilon}{n+1},\qquad\text{if }\boldsymbol{p}\in E.\end{aligned} \tag{3.4}$$

   This is possible as the Law of Large Number guarantees that if $\boldsymbol{p}\in U_i$ then as $N\to\infty$, $\mathbb{P}_{\boldsymbol{p}}\left(\widehat{\boldsymbol{p}}^{(N)}\in U_i\right)\to 1$.

5. Roll the $\boldsymbol{p}$-die N times and compute $\widehat{\boldsymbol{p}}^{(N)}$.

6. Consider a set of candidate outputs: $O=\left\{k_i:\widehat{\boldsymbol{p}}^{(N)}\in F_i\right\}$ and notice that it may be the null set or have cardinality greater than 1. If O is the empty set,

then output any arbitrary value in $\mathbb{N}$. Otherwise, output an arbitrary value in O.

We now show that the above algorithm satisfies eq. (3.3). Denote by X the output of the algorithm. After having found t in the first step, the algorithm can output a value in $\{k_i\}_{i=1,\dots,n}$or another arbitrary value, here denoted by z, if $O = \emptyset$.

- If $\boldsymbol{p} \in E$, we then have:

$$\mathbb{P}_{\boldsymbol{p}}\left(X = k_i, \boldsymbol{p} \notin U_{k_i}\right) \leq \mathbb{P}_{\boldsymbol{p}}\left(\widehat{\boldsymbol{p}}^{(N)} \in F_i, \boldsymbol{p} \notin U_{k_i}\right) < \frac{\epsilon}{n+1},$$

  as for X to be equal to k_i, it must be $\widehat{\boldsymbol{p}}^{(N)} \in F_i$ (notice however that it is not an equality as even if $\widehat{\boldsymbol{p}}^{(N)} \in F_i$, the algorithm may output something different than k_i as $\widehat{\boldsymbol{p}}^{(N)}$ may belong to multiple sets in $F_1, \dots, F_n$). Moreover:

$$\mathbb{P}_{\boldsymbol{p}}\left(X = z, \boldsymbol{p} \notin U_z\right) = \mathbb{P}_{\boldsymbol{p}}\left(\widehat{\boldsymbol{p}}^{(N)} \notin E, \boldsymbol{p} \notin U_z\right) < \mathbb{P}_{\boldsymbol{p}}\left(\widehat{\boldsymbol{p}}^{(N)} \notin E\right) < \frac{\epsilon}{n+1}.$$

  Therefore:

$$\begin{aligned}\sum_{\alpha:\boldsymbol{p}\notin U_\alpha} \mathbb{P}_{\boldsymbol{p}}\left(X = \alpha\right) &< \mathbb{P}_{\boldsymbol{p}}\left(X = z, \boldsymbol{p} \notin U_z\right) + \sum_{i=1}^{n} \mathbb{P}_{\boldsymbol{p}}\left(X = k_i, \boldsymbol{p} \notin U_{k_i}\right) \\ &< \frac{\epsilon}{n+1} + n\frac{\epsilon}{n+1} = \epsilon < \boldsymbol{p}^k.\end{aligned}$$

- If $\boldsymbol{p} \notin E$, it is enough to show that $\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{p} \notin E) < \boldsymbol{p}^k$. Recall that t is the number of rolls required to observe at least $(m+1)k+1$ rolls of each face. A string of length t, where each character is in $0, \dots, m$, with at least $(m+1)k+1$ '0' symbols in it, has probability at most $p_0^{(m+1)k+1}$ of occurring. Since there are m^t possible strings of length m, if $p_0 < m^{-t}$ we can upper bound the probability of observing at least $(m+1)k+1$ '0' by:

$$m^t p_0^{(m+1)k+1} < p_0^{(m+1)k} < \boldsymbol{p}^k.$$

  We can repeat the same reasoning for all the other possible outcomes, thus reaching the desired result.

We finally briefly discuss the case where S is a proper subset of Δ^m. The definition of the set E is left unchanged and it is determined at step 1 of the algorithm. However, E may not be a subset of S, so in the step 2 it may not be possible to consider a finite cover for E. Therefore we add the artificial set $H := \Delta^m \setminus S$ to the given

cover and we adapt N so that eq. (3.4) holds when considering $\frac{\epsilon}{n+2}$ as constant and moreover it holds:

$$\mathbb{P}_{\boldsymbol{p}}\left(\widehat{\boldsymbol{p}}^{(N)} \in H\right) < \frac{\epsilon}{n+2},$$

where this is again possible thanks to the Law of Large Numbers, since $\boldsymbol{p} \in S$. The algorithm then proceeds analogously, however if it would output the index corresponding to the artificial set H, we output any arbitrary k_i instead. □

Lemma 3.14. *Given a continuous function $f : S \subset \Delta^m \to (0,1)$ and a positive integer k, there exists a function $g : S \subset \Delta^m \to (0,1)$ such that for every $\boldsymbol{p} \in S$ there exists a Dice Enterprise for g and*

$$|f(\boldsymbol{p}) - g(\boldsymbol{p})| < \boldsymbol{p}^k, \qquad \forall \boldsymbol{p} \in S$$

Proof. Since f is continuous, we can choose an open cover $\{U_i\}_{i\in\mathbb{N}}$ of S and numbers $\{q_i\}_{i\in\mathbb{N}}$ so that:

$$|f(\boldsymbol{p}) - q_i| < \boldsymbol{p}^{k+1}, \qquad \boldsymbol{p} \in U_i. \tag{3.5}$$

Run the algorithm of Lemma 3.13 (using $k+1$ as constant) and denote its output by X, so that:

$$\sum_{\alpha:\boldsymbol{p}\notin U_\alpha} \mathbb{P}_{\boldsymbol{p}}(X = \alpha) < \boldsymbol{p}^{k+1}. \tag{3.6}$$

Let

$$g(\boldsymbol{p}) = \sum_{\alpha=1}^{\infty} q_\alpha \mathbb{P}_{\boldsymbol{p}}(X = \alpha).$$

We can toss a $g(\boldsymbol{p})$-coin by first simulating X and then tossing a q_α-coin, which we can do by just rolling the $\boldsymbol{p}$-die (cf. Section 1.4). Notice that using eq. (3.5) and (3.6) we reach the required result for every $\boldsymbol{p} \in S$:

$$\begin{aligned}
|f(\boldsymbol{p}) - g(\boldsymbol{p})| = \left|\sum_{\alpha=1}^{\infty}(f(\boldsymbol{p}) - q_\alpha)\mathbb{P}_{\boldsymbol{p}}(X=\alpha)\right| &\leq \sum_{\alpha=1}^{\infty} |f(\boldsymbol{p}) - q_\alpha|\mathbb{P}_{\boldsymbol{p}}(X=\alpha) \\
&\leq \sum_{\alpha:\boldsymbol{p}\in U_\alpha} |f(\boldsymbol{p}) - q_\alpha|\mathbb{P}_{\boldsymbol{p}}(X=\alpha) + \sum_{\alpha:\boldsymbol{p}\notin U_\alpha}^{\infty} |f(\boldsymbol{p}) - q_\alpha|\mathbb{P}_{\boldsymbol{p}}(X=\alpha) \\
&< \sum_{\alpha:\boldsymbol{p}\in U_\alpha}^{\infty} \boldsymbol{p}^{k+1}\mathbb{P}_{\boldsymbol{p}}(X=\alpha) + \sum_{\alpha:\boldsymbol{p}\notin U_\alpha}^{\infty} \mathbb{P}_{\boldsymbol{p}}(X=\alpha) \\
&\leq 2\boldsymbol{p}^{k+1} \leq \boldsymbol{p}^k.
\end{aligned}$$

□

Lemma 3.15. *There exists a Dice Enterprise for a function $f : S \subset \Delta^m \to (0,1)$ if f is continuous and there exists a positive integer k such that:*

$$\boldsymbol{p}^k < f(\boldsymbol{p}) < 1 - \boldsymbol{p}^k, \qquad \forall \boldsymbol{p} \in S. \tag{3.7}$$

Proof. We shall write $f(\boldsymbol{p})$ as

$$f(\boldsymbol{p}) = \sum_{n=1}^{\infty} \frac{1}{2^n} f_n(\boldsymbol{p}).$$

We can then toss an $f(\boldsymbol{p})$-coin by first simulating $K \sim \text{Geom}(1/2)$ and then tossing an $f_K(\boldsymbol{p})$-coin. A sample from such geometric distribution can be constructed given the $\boldsymbol{p}$-die by tossing a fair coin (cf. Section 1.4) until heads is observed. We are left to define functions $f_n(\boldsymbol{p})$ for which a Dice Enterprise must exist.

By Lemma 3.14 (using $k+1$ as constant) there exists a function f_1 such that

$$|f_1(\boldsymbol{p}) - f(\boldsymbol{p})| < \boldsymbol{p}^{k+1}, \qquad \forall p \in S,$$

and a Dice Enterprise for f_1 exists. Therefore $f_1(\boldsymbol{p}) > f(\boldsymbol{p}) - \boldsymbol{p}^{k+1}$ and by the assumption of eq. (3.7) we have:

$$\begin{aligned} f_1(\boldsymbol{p}) &> f(\boldsymbol{p}) - \boldsymbol{p}^{k+1} = 2f(\boldsymbol{p}) - f(\boldsymbol{p}) - \boldsymbol{p}^{k+1} \\ &> 2f(\boldsymbol{p}) - (1 - \boldsymbol{p}^k) - \boldsymbol{p}^{k+1} \\ &> 2f(\boldsymbol{p}) - 1 + \boldsymbol{p}^{k+1}. \end{aligned}$$

Analogously:

$$f_1(\boldsymbol{p}) < 2f(\boldsymbol{p}) - \boldsymbol{p}^{k+1},$$

so that by combining the two inequalities we have:

$$\boldsymbol{p}^{k+1} < 2f(\boldsymbol{p}) - f_1(\boldsymbol{p}) < 1 - \boldsymbol{p}^{k+1}.$$

Notice that $2f(\boldsymbol{p}) - f_1(\boldsymbol{p})$ is continuous (because $f_1(\boldsymbol{p})$ is continuous since it admits a Dice Enterprise, cf. Lemma 3.16) and polynomially bounded as in eq. (3.7) with constant $k+1$. Therefore we can reapply Lemma 3.14 (this time using $k+2$ as constant) and proceeds analogously as before to find a function f_2 such that a Dice Enterprise for f_2 exists and

$$\boldsymbol{p}^{k+2} < 4f(\boldsymbol{p}) - 2f_1(\boldsymbol{p}) - f_2(\boldsymbol{p}) < 1 - \boldsymbol{p}^{k+2}.$$

Continuing this way we can find functions $f_1, f_2, \ldots$ that admit a Dice Enterprise and such that for every n we have

$$\boldsymbol{p} < 2^n f(\boldsymbol{p}) - 2^{n-1} f_1(\boldsymbol{p}) - \ldots - f_n(\boldsymbol{p}) < 1 - \boldsymbol{p}.$$

Replacing the two bounds by 0 and 1 respectively and dividing by 2^n we get

$$0 < f(\boldsymbol{p}) - \frac{1}{2} f_1(\boldsymbol{p}) - \ldots - \frac{1}{2^n} f_n(\boldsymbol{p}) < \frac{1}{2^n},$$

so that indeed as $n \to \infty$ we have $f(\boldsymbol{p}) = \sum_{n=1}^{\infty} \frac{1}{2^n} f_n(\boldsymbol{p})$, as desired. □

Lemma 3.16. *Given $f : S \subset \Delta^m \to \Delta^v$, if a Dice Enterprise for f exists then:*

- *Each $f_i(\boldsymbol{p})$ is continuous;*
- *There exists $n_0 \in \mathbb{N}$ such that each $f_i(\boldsymbol{p})$ satisfies $f_i(\boldsymbol{p}) \geq \min(p_0, \ldots, p_m)^{n_0}$ for all $\boldsymbol{p} \in S$.*

Proof. Continuity: having fixed $\boldsymbol{q} \in S$, we shall prove that $\forall \epsilon > 0, \exists \delta$ s.t. $\forall \boldsymbol{p} \in S$ with $|\boldsymbol{p} - \boldsymbol{q}| < \delta$, then $|f_i(\boldsymbol{p}) - f_i(\boldsymbol{q})| < \epsilon$. Fix ϵ and let T be the (random) number of iterations required for the algorithm to terminate. Since it ends almost surely, there exists n such that $\mathbb{P}_{\boldsymbol{p}}(T \leq n) > 1 - \frac{\epsilon}{2}$. Moreover, following the notation of Theorem 3.9, denote by $\mathbb{P}_{\boldsymbol{p}}(\text{algorithm outputs } i, T \leq n) = f_i^{(n)}(\boldsymbol{p})$. Note that $f_i^{(n)}(\boldsymbol{p})$ is a polynomial, therefore continuous and such that $|f_i^{(n)}(\boldsymbol{p}) - f_i^{(n)}(\boldsymbol{q})| < \frac{\epsilon}{2}$ for a δ with $|\boldsymbol{p} - \boldsymbol{q}| < \delta$. Then notice:

$$\begin{aligned} f_i(\boldsymbol{p}) \geq f_i^{(n)}(\boldsymbol{p}) &> f_i^{(n)}(\boldsymbol{q}) - \frac{\epsilon}{2} = \mathbb{P}_{\boldsymbol{q}}(\text{algorithm outputs } i, T \leq n) - \frac{\epsilon}{2} \\ &= f_i(\boldsymbol{q}) + \mathbb{P}_{\boldsymbol{p}}(T \leq n) - \mathbb{P}_{\boldsymbol{q}}(\text{algorithm outputs } i \text{ or } T \leq n) - \frac{\epsilon}{2} \\ &\geq f_i(\boldsymbol{q}) + \mathbb{P}_{\boldsymbol{p}}(T \leq n) - 1 - \frac{\epsilon}{2} \geq f_i(\boldsymbol{q}) - \epsilon \end{aligned}$$

and analogously denote $f_{-i}^{(n)}(\boldsymbol{p}) = \sum_{j \neq i} f_i^{(n)}(\boldsymbol{p})$ and notice that it is continuous, so by the same argument:

$$1 - f_i(\boldsymbol{p}) \geq f_{-i}^{(n)}(\boldsymbol{p}) \geq f_{-i}^{(n)}(\boldsymbol{q}) - \frac{\epsilon}{2} = 1 - f_i(\boldsymbol{q}) - \epsilon$$

and thus $f_i(\boldsymbol{p}) < f_i(\boldsymbol{q}) + \epsilon$.

Polynomially bounded: Following the notation of the proof of Theorem 3.9, let n_0 be the smallest n such that $A_i^{(n)}$ is not empty. Notice that there cannot be an i such that $A_i^{(n)} = \emptyset$ for all n, as we have proved that $\lim_{n \to \infty} \mathbb{P}\left(A_i^{(n)}\right) = f_i^{(n)}(p) > 0$. The

desired result follows by noticing:

$$f_i(p) \geq \mathbb{P}\left(A_i^{(n_0)}\right) = \mathbb{P}\left(\bigcup_{\boldsymbol{k}\in\Lambda_{n_0}^m} A_{i,\boldsymbol{k}}^{(n_0)}\right) = \sum_{\boldsymbol{k}\in\Lambda_{n_0}^m} \left|A_{i,\boldsymbol{k}}^{(n_0)}\right| \boldsymbol{p}^{\boldsymbol{k}} \geq \boldsymbol{p}^{\boldsymbol{k}^\star},$$

for a $\boldsymbol{k}^\star$ such that $\left|A_{i,\boldsymbol{k}^\star}^{(n_0)}\right| > 0$. Finally, notice that $\left\|\boldsymbol{k}^\star\right\|_1 = n_0$, so that:

$$f_i(p) \geq \boldsymbol{p}^{\boldsymbol{k}^\star} \geq \min(p_0^{n_0}, \ldots, p_m^{n_0}),$$

as desired. □

Theorem 3.17. *Given $f : S \subset \Delta^m \to \Delta^v$, a Dice Enterprise for f exists if and only if:*

- *Each $f_i(\boldsymbol{p})$ is continuous;*
- *There exists $n_0 \in \mathbb{N}$ such that each $f_i(\boldsymbol{p})$ satisfies $f_i(\boldsymbol{p}) \geq \min(p_0, \ldots, p_m)^{n_0}$ for all $\boldsymbol{p} \in S$.*

Proof. Sufficiency. We first notice that the second assumption implies that for each $i \in \{0, \ldots, v\}$ there exists $k \in \mathbb{N}$ such that $\boldsymbol{p}^k < f_i(\boldsymbol{p}) < 1 - \boldsymbol{p}^k$. For brevity, denote $q := \min(p_0, \ldots, p_m)$ and notice that:

$$f_i(\boldsymbol{p}) \geq q^{n_0} > \boldsymbol{p}^{n_0},$$

where recall that $\boldsymbol{p}^{n_0}$ denotes $p_0^{n_0} \cdot \ldots \cdot p_m^{n_0}$. Similarly:

$$1 - f_i(\boldsymbol{p}) = \sum_{j=0, j\neq i}^{v} f_j(\boldsymbol{p}) > vq^{n_0} > v\boldsymbol{p}^{n_0},$$

so that there exists k such that $v\boldsymbol{p}^{n_0} > \boldsymbol{p}^k$. We have proved in Lemma 3.15 that under the required assumptions there exists a Dice Enterprise for each $f_i(\boldsymbol{p})$ and we can therefore toss f_i-coins. The desired result follows by applying Lemma 3.8.
Necessity. Proved in Lemma 3.16. □

3.4 Fast Simulation

In Section 3.3, we established for which class of functions a Dice Enterprise exists. When polynomial approximations of a function $f(\boldsymbol{p})$ are available as in Theorem 3.9,

the distribution of the number of rolls N of the $\boldsymbol{p}$-die is known. However, analysing the law of N for generic functions which admit a Dice Enterprise is challenging. In particular, we are interested in finding a class of functions for which a fast simulation is available, in the sense of Nacu and Peres [43]:

Definition 3.18 (Fast simulation)**.** A Dice Enterprise for a function $f(\boldsymbol{p}) : S \subset \Delta^m \to \Delta^v$ has a *fast simulation* if for any $\boldsymbol{p} \in S$ there exist constants $C > 0$, $\rho < 1$, which may depend on $\boldsymbol{p}$, such that the number N of rolls of the $\boldsymbol{p}$-die required for the algorithm to terminate satisfies $\mathbb{P}_{\boldsymbol{p}}(N > n) \leq C\rho^n$.

In this section we provide the necessary results to prove a generalisation of Theorem 2 of Nacu and Peres [43] and show that, under standard assumptions, a Dice Enterprise with a fast simulation exists if and only if the function f is real analytic. It is worth clarifying what a real analytic function is in our context. In particular, define the m dimensional open disk as:

$$D^m = \left\{ \boldsymbol{p} = (p_1, \dots, p_m) \in (0,1)^m : \sum_{i=1}^m p_i < 1 \right\}.$$

Notice that in this instance we purposefully let the indices start from 1 rather than 0. This allows to easily define a bijection $\psi : \Delta^m \to D^m$ as:

$$\begin{aligned} &\psi(\boldsymbol{p}) = \psi(p_0, \dots, p_m) = (p_1, \dots, p_m), \\ &\psi^{-1}(\boldsymbol{p}) = \psi^{-1}(p_1, \dots, p_m) = \left(1 - \sum_{i=1}^m p_i, p_1, \dots, p_m\right). \end{aligned}$$

Therefore, given a function $f : \Delta^m \to \Delta^v$, we can equivalently consider $\widetilde{f} : D^m \to \Delta^v$ by setting $\widetilde{f}(\boldsymbol{p}) = f(\psi(\boldsymbol{p}))$. We will slightly abuse notation and just consider functions $f : D^m \to \Delta^v$ and still refer to a $\boldsymbol{p}$-die.

We also review accordingly the scaled by n discrete m dimensional simplex Λ_n^m and define:

$$\Theta_n^m = \left\{ \boldsymbol{k} = (k_1, \dots, k_m) \in \{0, 1, \dots, n\}^m : \sum_{i=1}^m k_i = n \right\},$$

which is equivalent to Δ_n^{m-1}, but we use a different notation to highlight that the indices now start from 1. A function $f : D^m \to \Delta^v$ is *real analytic* on an open subset $I \subset D^m$ if for each $\boldsymbol{p} \in I$ all $f_i(\boldsymbol{p})$ can be represented by an absolutely convergent

power series in a neighbourhood of $\boldsymbol{p}$, that is:

$$f_i(\boldsymbol{p}) = \sum_{\boldsymbol{k} \in \mathbb{N}^m} a_{\boldsymbol{k}} (\boldsymbol{p} - \boldsymbol{c})^{\boldsymbol{k}},$$

where $\boldsymbol{c} \in \mathbb{R}^m$ and $a_{\boldsymbol{k}} \in \mathbb{R}$ for each $\boldsymbol{k} \in \mathbb{N}^m$. We consider series that converge absolutely so that the order of summation does not matter. It is also worth pointing out that if f is real analytic in a point $\boldsymbol{q} \in D^m$, than it is also analytic for all $\boldsymbol{p} \in D^m$ such that $\boldsymbol{p} \leq \boldsymbol{q}$ (see e.g. Proposition 2.1.7 of [31]).

At the end of this section we will be able to prove the following result, that clearly defines the class of functions for which a Dice Enterprise with a fast simulation exists:

Theorem 3.19. *Let S be an open subset of D^m and consider $f(\boldsymbol{p}) : S \subset D^m \to \Delta^v$. Assume f is real analytic on the closed set $I = [a_1, b_1] \times \cdots \times [a_m, b_m] \subset S$, then there exists a Dice Enterprise for f that has a fast simulation on I. Vice-versa, if there is a Dice Enterprise for f that has a fast simulation, then f is real analytic on S.*

It is still an open question whether the above result holds for generic closed sets $I \subset S$. We shall proceed to prove the result as follows:

1. We prove that a Dice Enterprise with a fast simulation exists for functions admitting positive power series centred at the origin. Importantly, our proof is constructive and we provide an efficient algorithm, characterising its running time. It makes use of the linear Bernoulli Factory we introduced in Section 2.2.

2. We then provide the result for power series – not necessarily positive – centred at the origin and use such result to prove the first implication of Theorem 3.19.

3. By considering a suitable map from the compact set I to $(0, 1)^m$ that preserves fast simulation, we prove the first implication of Theorem 3.19. We then conclude by proving that if a Dice Enterprise has a fast simulation, then f must be real analytic.

Most of the results borrow from the ideas of Nacu and Peres [43]. In the following we will make use of a lemma on random variables with exponential tails.

Lemma 3.20 (Propositions 11,12 [43])**.** *Let X be a non-negative random variable. Then the following are equivalent:*

- *There exist constants $C > 0$, $\rho < 1$ such that $\mathbb{P}(X > x) \leq C\rho^x, \forall x > 0$;*

- $\mathbb{E}[\exp(tX)] < \infty$ *for some* $t > 0$.

If these hold we say that X *has exponential tail. Moreover, let* X_i *be i.i.d. random variables with exponential tails and* $N \geq 0$ *an integer valued random variable with exponential tail. Then* $Y = X_1 + \ldots + X_N$ *has exponential tail.*

3.4.1 Fast simulation for analytic functions

Write $f(\boldsymbol{p}) = (f_0(\boldsymbol{p}), \ldots, f_v(\boldsymbol{p}))$ where $f_i(\boldsymbol{p}) : \Delta^m \to (0,1)$. We first show that if a Dice Enterprise for $f_i(\boldsymbol{p})$ has a fast simulation for all $i = 0, \ldots, v$, then so does a Dice Enterprise for $f(\boldsymbol{p})$. Therefore, we can simplify the analysis and just study functions with image in $(0,1)$.

Proposition 3.21. *There exists a Dice Enterprise with a fast simulation of* $f(\boldsymbol{p}) = (f_0(\boldsymbol{p}), \ldots, f_v(\boldsymbol{p})) : S \subset \Delta^m \to \Delta^v$ *if and only if there exist Dice Enterprises with a fast simulation for each* $f_i(\boldsymbol{p})$, $i \in \{0, \ldots, v\}$.

Proof. $\Rightarrow$ A Dice Enterprise for $f_i(\boldsymbol{p})$ can be obtained by rolling the $f(\boldsymbol{p})$-die once and output 1 if it lands on the i^{th} face. The statement holds since by hypothesis the Dice Enterprise for $f(\boldsymbol{p})$ has a fast simulation.
$\Leftarrow$ We construct a Dice Enterprise for $f(\boldsymbol{p})$ via a small modification of the algorithm described in the proof of Lemma 3.8 and show that it has a fast simulation. At each iteration t we obtain a toss of all $f_i(\boldsymbol{p})$-coins and let N_t be the total number of rolls of the $\boldsymbol{p}$-die required. We then uniformly at random select $i \in \{0, \ldots, v\}$ and output i if the toss of the $f_i(\boldsymbol{p})$-coin resulted in heads, restarting otherwise. Then, the number N of rolls required by such algorithm is given by $N = N_1 + \ldots N_M$, where $M \sim \text{Geom}\left(\frac{1}{v+1}\right)$. Note that N_t are i.i.d. random variables with exponential tails and the required result follows by Lemma 3.20. □

We now explicitly provide in Algorithm 14 a Dice Enterprise for functions f that admit a non-negative power series expansion centred at the origin.

Proposition 3.22. *Let* $f(\boldsymbol{p}) : S \subset D^m \to (0,1)$ *where* $f(\boldsymbol{p}) = \sum_{\boldsymbol{k} \in \mathbb{N}^m} a_{\boldsymbol{k}} \boldsymbol{p}^{\boldsymbol{k}}$ *with* $a_{\boldsymbol{k}} \geq 0$ *for all* $\boldsymbol{k} \in \mathbb{N}^m$. *Let* $\boldsymbol{t}$ *be in the domain of convergence with* $0 < \sum_{\boldsymbol{k} \in \mathbb{N}^m} a_{\boldsymbol{k}} \boldsymbol{t}^{\boldsymbol{k}} \leq 1$. *If* $\boldsymbol{t} = \boldsymbol{1}$ *or* $\boldsymbol{t} > \boldsymbol{p}, \forall \boldsymbol{p} \in \bar{S}$ *then there exists a Dice Enterprise for* f *that has a fast simulation on* S. *Otherwise, it has a fast simulation on* $\{\boldsymbol{p} \in S : \boldsymbol{p} \leq \boldsymbol{t} - \epsilon\}, \forall \epsilon > 0$.

Proof. The main idea is to write $f(\boldsymbol{p})$ equivalently as:

$$f(\boldsymbol{p}) = \sum_{\boldsymbol{k} \in \mathbb{N}^m} a_{\boldsymbol{k}} \boldsymbol{p}^{\boldsymbol{k}} = f(\boldsymbol{t}) \sum_{i=0}^{\infty} \sum_{\boldsymbol{k} \in \Theta_i^m} \frac{a_{\boldsymbol{k}} \boldsymbol{t}^{\boldsymbol{k}}}{f(\boldsymbol{t})} \left(\frac{\boldsymbol{p}}{\boldsymbol{t}}\right)^{\boldsymbol{k}},$$

and noticing that Algorithm 14 outputs 1 accordingly. Indeed, let Z be the final output of the algorithm, then:

$$\begin{aligned}\mathbb{P}(Z=1) &= \mathbb{P}(X=1)\sum_{i=0}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m}\mathbb{P}(Z=1|Y=i,\boldsymbol{K}=\boldsymbol{k})\mathbb{P}(\boldsymbol{K}=\boldsymbol{k}|Y=i)\mathbb{P}(Y=i)\\ &= f(\boldsymbol{t})\sum_{i=0}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m}\left(\frac{\boldsymbol{p}}{\boldsymbol{t}}\right)^{\boldsymbol{k}}\frac{a_{\boldsymbol{k}}\boldsymbol{t}^{\boldsymbol{k}}}{c_i}\frac{c_i}{f(\boldsymbol{t})} = f(\boldsymbol{p}),\end{aligned}$$

where $c_i = \sum_{\boldsymbol{k}\in\Theta_i^m} a_{\boldsymbol{k}}\boldsymbol{t}^{\boldsymbol{k}}$, which is finite as $f(\boldsymbol{t}) \leq 1$. It is left to check that the sampling steps at lines 2, 4, 5, 6 of the algorithm are valid and that the procedure has a fast simulation. Note that by hypothesis, $f(\boldsymbol{t})$ is a constant in $(0,1]$ and since all the coefficients $a_{\boldsymbol{k}}$ are non-negative, the distributions of Y and $\boldsymbol{K}$ are well defined. We can then sample X, Y, and $\boldsymbol{K}$ via a Dice Enterprise for constant dice, which has a fast simulation (cf. Section 1.4). If $t_i \geq 1$, we can toss a p_i/t_i-coin by tossing a p_i-coin and a $(1/t_i)$-coin, outputting heads if both tosses result in heads. Otherwise, if $\boldsymbol{t} > \boldsymbol{p}, \forall \boldsymbol{p} \in \bar{S}$, there must exist $\epsilon > 0$ such that $\boldsymbol{p} \leq \boldsymbol{t} - \epsilon$ for all $\boldsymbol{p} \in S$. Then:

$$\frac{p_i}{t_i} \leq \frac{t_i - \epsilon}{t_i} \leq 1 - \frac{\epsilon}{t_i},$$

and we can use a linear Bernoulli Factory to toss a (p_i/t_i)-coin, which has a fast simulation (cf. Section 2.2). We then conclude that also Algorithm 14 has a fast simulation, in light of Lemma 3.20. Notice that although the lemma does not apply directly, we can resort to a trick similar to that of the proof of Proposition 3.21 to prove that the algorithm has indeed a fast simulation. □

Remark 3.23. If we assume $f(\boldsymbol{p}) \leq 1 - \epsilon$ for a known $\epsilon > 0$, we can drop the hypothesis $f(\boldsymbol{t}) \leq 1$. Indeed, we can resort to Algorithm 14 and skip line 2, thus producing tosses of a coin with associated probability equal to $f(\boldsymbol{p})/f(\boldsymbol{t})$. We can finally resort to a linear Bernoulli factory to get tosses of an $f(\boldsymbol{p})$-coin.

Algorithm 14 Dice Enterprise for Positive Power Series Centred at $\mathbf{0}$ as in Proposition 3.22

1: Denote $c_i := \sum_{\boldsymbol{k} \in \Theta_i^m} a_{\boldsymbol{k}} \boldsymbol{t}^{\boldsymbol{k}}$
2: Let $X \sim \text{Bern}\left(f(\boldsymbol{t})\right)$
3: **if** $X = 0$ **then** stop and **return** 0
4: Sample $Y \in \mathbb{N}$ such that $\mathbb{P}(Y = i) = \frac{c_i}{f(\boldsymbol{t})}$
5: Sample $\boldsymbol{K} \in \Theta_Y^m$ such that $\mathbb{P}(\boldsymbol{K} = \boldsymbol{k}) = \frac{a_{\boldsymbol{k}} t^{\boldsymbol{k}}}{c_Y}$
6: Toss a $\frac{p_1}{t_1}$-coin K_1 times,…,$\frac{p_m}{t_m}$-coin K_m times (cf. Section 2.2). Stop prematurely if a toss results in tails.
7: **if** all previous tosses resulted in heads **then return** 1 **else return** 0

The algorithm we have just proposed and the result we prove in Proposition 3.22 is not a mere generalisation of the analogous results of Nacu and Peres [43] (Proposition 16). Even in the Bernoulli Factory case, our construction is valid for a wider variety of functions f as it allows for the case $f(\boldsymbol{t}) = 1$ and the Dice Enterprise has a fast simulation on the whole set S if $\boldsymbol{t} > \boldsymbol{p}, \forall \boldsymbol{p} \in \bar{S}$. Moreover, our algorithm has an efficient implementation, as we now show in the following corollary.

Corollary 3.24. *Let $f(\boldsymbol{p})$ as in Proposition 3.22 and consider $\bar{p} := \max_{i=1,\dots,m} p_i$ and $\underline{t} := \min_{i=1,\dots,m} t_i$ Let $\bar{\boldsymbol{p}} = \bar{p}\mathbf{1}$ (i.e. an m-long vector with the maximum of $\boldsymbol{p}$ in all its entries) and N be the number of $\boldsymbol{p}$-die rolls required by Algorithm 14, then:*

$$\mathbb{E}[N] \leq \begin{cases} \dfrac{f(\boldsymbol{t}) - f(\bar{\boldsymbol{p}})}{\underline{t} - \bar{p}} & \text{if } \underline{t} \geq 1 \\ 5.53(1 + \epsilon^{-1}) \cdot \dfrac{f(\boldsymbol{t}) - f(\bar{\boldsymbol{p}})}{\underline{t} - \bar{p}} & \text{if } \underline{t} < 1 \end{cases},$$

assuming a generator of uniform random variables is available.

Proof. We apply Algorithm 14 and resort to the generator of uniform random variables to sample X, Y, and $\boldsymbol{K}$. Note that the algorithm outputs 1 only if $X = 1$ and all the required (p_i/t_i)-coins tosses result in heads. Therefore, we can optimise N by prematurely stopping the algorithm as soon as tails is observed. We obtain an upper bound on N by considering the number of $\boldsymbol{p}$-die rolls required to toss a $(\bar{p}/\underline{t})^{k_i}$-coin rather than $(p_i/t_i)^{k_i}$-coin. Let N_i be the number of $(\bar{p}/\underline{t})$-coin tosses

required to toss a $(\bar{p}/\underline{t})^i$-coin. We obtain:

$$\mathbb{P}(N_i = n) = \begin{cases} \left(\frac{\bar{p}}{\underline{t}}\right)^{n-1}\left(1-\frac{\bar{p}}{\underline{t}}\right) & \text{if } 0 < n < i \\ \left(\frac{\bar{p}}{\underline{t}}\right)^{n-1} & \text{if } n = i \\ 0 & \text{if } n \leq 0 \text{ or } n > i \end{cases},$$

$$\mathbb{P}(N_i > n) = \begin{cases} 1 & \text{if } n < 0 \\ \left(\frac{\bar{p}}{\underline{t}}\right)^{n} & \text{if } 0 \leq n < i \\ 0 & \text{if } n \geq i \end{cases}.$$

Let $N_{\bar{\boldsymbol{p}}/\boldsymbol{t}}$ be the number of tosses of a $(\bar{p}/\underline{t})$-coin required by the algorithm, so that for $n > 0$:

$$\begin{aligned} \mathbb{P}(N_{\bar{p}/\underline{t}} > n) &\leq \mathbb{P}(X = 1)\sum_{i=0}^{\infty}\mathbb{P}(N_i > n)\mathbb{P}(Y = i) \\ &= f(\boldsymbol{t})\sum_{i=0}^{\infty}\mathbb{P}(N_i > n)\frac{1}{f(\boldsymbol{t})}\sum_{\boldsymbol{k}\in\Theta_i^m} a_{\boldsymbol{k}}\boldsymbol{t}^{\boldsymbol{k}} \\ &= \sum_{i=n+1}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m} a_{\boldsymbol{k}}\boldsymbol{t}^{\boldsymbol{k}}\left(\frac{\bar{p}}{\underline{t}}\right)^n. \end{aligned}$$

Therefore, we can upper bound the expected value of $N_{\bar{p}/\underline{t}}$ by noticing $\boldsymbol{t}^{\boldsymbol{k}} \geq \underline{t}^{\|\boldsymbol{k}\|_1}$, thus obtaining:

$$\begin{aligned} \mathbb{E}[N_{\bar{p}/\underline{t}}] &\leq \sum_{n=0}^{\infty}\sum_{i=n+1}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m} a_{\boldsymbol{k}}\boldsymbol{t}^{\boldsymbol{k}}\left(\frac{\bar{p}}{\underline{t}}\right)^n = \sum_{i=1}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m}\sum_{n=0}^{i-1} a_{\boldsymbol{k}}\boldsymbol{t}^{\boldsymbol{k}}\left(\frac{\bar{p}}{\underline{t}}\right)^n \\ &= \sum_{i=1}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m} a_{\boldsymbol{k}}\boldsymbol{t}^{\boldsymbol{k}}\frac{1-(\bar{p}/\underline{t})^i}{1-\bar{p}/\underline{t}} \\ &= \frac{1}{1-\bar{p}/\underline{t}}\left[\sum_{i=1}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m} a_{\boldsymbol{k}}\boldsymbol{t}^{\boldsymbol{k}} - \sum_{i=1}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m} a_{\boldsymbol{k}}\boldsymbol{t}^{\boldsymbol{k}}\left(\frac{\bar{p}}{\underline{t}}\right)^i\right] \\ &\leq \frac{1}{1-\bar{p}/\underline{t}}\left[\sum_{i=0}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m} a_{\boldsymbol{k}}\boldsymbol{t}^{\boldsymbol{k}} - \sum_{i=0}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m} a_{\boldsymbol{k}}\underline{t}^i\left(\frac{\bar{p}}{\underline{t}}\right)^i\right] \\ &= \frac{1}{1-\bar{p}/\underline{t}}\left[f(\boldsymbol{t}) - \sum_{i=0}^{\infty}\sum_{\boldsymbol{k}\in\Theta_i^m} a_{\boldsymbol{k}}\bar{\boldsymbol{p}}^{\boldsymbol{k}}\right] = \frac{f(\boldsymbol{t}) - f(\bar{\boldsymbol{p}})}{1-\bar{p}/\underline{t}}. \end{aligned}$$

In the case $\underline{t} \geq 1$, we can toss a $(\bar{p}/\underline{t})$-coin by first simulating $U \sim \text{Unif}(0,1)$ and outputting heads only if $U \leq 1/\underline{t}$ and a $\bar{p}$-coin toss results in heads. Therefore, there is need to toss a $\bar{p}$-coin only if $U \leq 1/\underline{t}$ and the expected number of $\bar{p}$-coin tosses is $1/\underline{t}$. Since, tossing a $\bar{p}$-coin requires one roll of the $\boldsymbol{p}$-die, we obtain the desired result. On the other hand, if $\underline{t} < 1$ we need to resort to a linear Bernoulli Factory to toss a $(\bar{p}/\underline{t})$-coin, cf. Section 2.2. Since we consider $\boldsymbol{p}$ such that $\bar{p}/\underline{t} \leq 1 - \epsilon/\underline{t}$ the expected number of p_i-coin tosses required is smaller than $5.53\underline{t}(1+\epsilon^{-1})$, leading to the required result.

□

An efficient algorithm for positive power series centred at 0 in the Bernoulli Factory case had already been proposed by Mendo [38] under stricter assumptions, namely $\lim_{p\to 1} f(p) = 1$. Under the same assumptions, our proposed construction is equivalent to that of Mendo [38] in terms of computational cost (measured as the number of p-coin tosses required) as we can set $t = 1$. However, our result – besides being valid for the Dice Enterprise case rather than just for Bernoulli Factories – relaxes Mendo's hypothesis and still provides a tight upper bound on the number of $\boldsymbol{p}$-die rolls required.

Example 3.25. Assume a 3-sided die is given; as discussed at the beginning of Section 3.4 we consider analogously the vector $\boldsymbol{p} = (p_1, p_2)$ representing the probabilities of rolling the 1^{st} and 2^{nd} face. Clearly the probability of rolling the 0^{th} face is given by $1 - p_1 - p_2$. Consider the function:

$$f(\boldsymbol{p}) := \frac{p_1}{1 - p_1 p_2} = \sum_{\boldsymbol{k}\in\mathbb{N}^2} \mathbb{I}\{k_1 = 1 + k_2\} p_1^{k_1} p_2^{k_2}.$$

Assume $\epsilon_1, \epsilon_2 > 0$ such that $p_i \leq 1 - \epsilon_i$ are known. Although the theory we developed just consider a single $\epsilon > 0$ for better clarity, from an implementation point of view it is more efficient to consider different values of ϵ and apply the strategies detailed in Proposition 3.22. We can set $\boldsymbol{t} = \left(1 - \frac{\epsilon_1}{2}, 1 - \frac{\epsilon_2}{2}\right)$, so that

$$\frac{p_i}{t_i} \leq 1 - \frac{\epsilon_i}{2 - \epsilon_i}.$$

If $f(\boldsymbol{t}) \leq 1$ we can directly apply Algorithm 14, otherwise we make use of Remark 3.23 and notice that

$$f(\boldsymbol{p}) \leq \frac{1 - \epsilon_1}{1 - (1 - \epsilon_1)(1 - \epsilon_2)}.$$

We run the algorithm for different choices of p_1, p_2. We consider the best possible

ϵ, i.e. setting $\epsilon_i = 1 - p_i$.

$\boldsymbol{p}$	$(0.1, 0.2)$	$(\sqrt{5}-2, \sqrt{5}-2)$	$(0.3, 0.6)$	$(0.9, 0.05)$
$f(\boldsymbol{p})$	0.102	0.245	0.363	0.942
$f(\boldsymbol{t})$	0.821	1.00	0.35	1.90
$\widehat{f}(\boldsymbol{p})$	$_{0.096}0.101_{\,0.108}$	$_{0.237}0.245_{\,0.254}$	$_{0.354}0.363_{\,0.373}$	$_{0.936}0.942_{\,0.945}$
$\widehat{\mathbb{E}}[N]$	5.3439	27.3961	57.2085	8774.94

Table 3: Implementation of a Dice Enterprise for $f(\boldsymbol{p}) = p_1/(1 - p_1p_2)$ and for different values of the true unknown probability $\boldsymbol{p}$. The algorithm has been run 10,000 times to obtain tosses of the $f(\boldsymbol{p})$-coin and $\widehat{f}(\boldsymbol{p})$ is the sample average. Smaller number represent 95% confidence intervals computed via the Wilson score interval. $\widehat{\mathbb{E}}[N]$ is the empirical average number of rolls of the $\boldsymbol{p}$-die required.

We now consider generic power series centred at the origin. The algorithm idea is simple: we split the series according to the sign of its coefficients and then resort to a Bernoulli Factory for coins subtraction (cf. Proposition 2.27).

Proposition 3.26. *Let $f(\boldsymbol{p}) : S \subset D^m \to (0,1)$ where $f(\boldsymbol{p}) = \sum_{\boldsymbol{k}\in\mathbb{N}^m} a_{\boldsymbol{k}}\boldsymbol{p}^{\boldsymbol{k}}$. Let $\boldsymbol{t}$ be in the domain of convergence with $f(\boldsymbol{t}) \leq 1$ and consider $I := \{\boldsymbol{p} \in S : \boldsymbol{p} \leq \boldsymbol{t} - \epsilon\}$ such that $\epsilon \leq f(\boldsymbol{p}) \leq 1 - \epsilon$, for all $\boldsymbol{p} \in I$. Then there exists a Dice Enterprise for f that has a fast simulation on I.*

Proof. We follow the general idea of the proof of Proposition 17 of Nacu and Peres [43]. We first separate the positive and negative coefficients, letting:

$$g(\boldsymbol{p}) := \sum_{\substack{\boldsymbol{k}\in\mathbb{N}^m \\ a_{\boldsymbol{k}}\geq 0}} a_{\boldsymbol{k}}\boldsymbol{p}^{\boldsymbol{k}}, \quad h(\boldsymbol{p}) := \sum_{\substack{\boldsymbol{k}\in\mathbb{N}^m \\ a_{\boldsymbol{k}}< 0}} |a_{\boldsymbol{k}}|\boldsymbol{p}^{\boldsymbol{k}}, \quad f(\boldsymbol{p}) = g(\boldsymbol{p}) - h(\boldsymbol{p}).$$

Let $M := \sum_{\boldsymbol{k}\in\mathbb{N}^m} |a_{\boldsymbol{k}}|\boldsymbol{t}^{\boldsymbol{k}}$ and notice that for $\boldsymbol{p} \in I$, $g(\boldsymbol{t})/M \leq 1$ and $h(\boldsymbol{t})/M \leq 1$. Therefore, we can apply Proposition 3.22 to obtain tosses of a $g(\boldsymbol{p})/M$-coin and an $h(\boldsymbol{p})/M$-coin. By assumption, $f(\boldsymbol{p})/M \geq \epsilon/M$ and we can then resort to Proposition 2.27 to obtain tosses of a $f(\boldsymbol{p})/M$-coin. Finally, we resort to a linear Bernoulli Factory (cf. Section 2.2) to get tosses of an $f(\boldsymbol{p})$-coin as $f(\boldsymbol{p}) \leq 1 - \epsilon$. □

Remark 3.27. If $M := \sum_{\boldsymbol{k}\in\mathbb{N}^m} |a_{\boldsymbol{k}}|\boldsymbol{t}^{\boldsymbol{k}} \leq 1$, the assumption $f(\boldsymbol{p}) \leq 1 - \epsilon$ can be dropped.

We now restrict the analysis to closed subset of D^m of the form $I = [a_1, b_1] \times \cdots \times [a_m, b_m]$ to prove the first implication of Theorem 3.19. We first point out in the following corollary that the assumptions of Proposition 3.26 are automatically

satisfied for $\boldsymbol{p} \in I$. We then proceed to prove that any analytic function has a fast simulation, the main idea being mapping the domain of the function to $(0,1)^m$ so that Proposition 3.26 can be applied.

Corollary 3.28. *Let S be an open subset of D^m and consider $f(\boldsymbol{p}) : S \to (0,1)$ where $f(\boldsymbol{p}) = \sum_{\boldsymbol{k} \in \mathbb{N}^m} a_{\boldsymbol{k}} \boldsymbol{p}^{\boldsymbol{k}}$. Consider $I := [a_1, b_1] \times \cdots \times [a_m, b_m] \subset S$. Then there exists a Dice Enterprise for f that has a fast simulation on I.*

Proof. We show that all the assumptions of Proposition 3.26 are satisfied. Indeed, since $I \subset S$ and S is an open set, one can consider $\boldsymbol{t} \in S \setminus I$ such that $\boldsymbol{t} > (b_1, \ldots, b_m)$ and $f(\boldsymbol{t}) < 1$. Finally, notice that there must exist ϵ such that $\epsilon \leq f(\boldsymbol{p}) \leq 1 - \epsilon$ as I is a compact set and $f(\boldsymbol{p}) \in (0,1)$ for all $\boldsymbol{p} \in I$. □

Lemma 3.29. *Let S be an open subset of D^m and consider $f(\boldsymbol{p}) : S \subset D^m \to (0,1)$. Assume f is real analytic on the closed set $I = [a_1, b_1] \times \cdots \times [a_m, b_m] \subset S$, then there exists a Dice Enterprise for f that has a fast simulation on I.*

Proof. We make use of the results detailed in the proof of Theorem 19 by Nacu and Peres [43]. Let f be real analytic on a domain E containing I. If E is the open disk of radius 1 and centred at the origin, that the result follows by Corollary 3.28. Otherwise, we map the domain E to $(0,1)^m$ using a map that has a fast simulation. The proof of Nacu and Peres [43] provides a map $g : [a,b] \to (0,1)$ that admits a Bernoulli Factory with a fast simulation. Moreover it guarantees that $f(g(p_0), \ldots, g(p_m))$ is real analytic on the open disk of radius 1 centred at the origin. As noticed, the domain of such function may not be a subset of D^m, but it is a subset of $[0,1]^m$ and the proof of Corollary 3.28 still holds in this case. It remains to check that function composition preserves fast simulation. This can be proved in the same manner of Proposition 14(ii) of Nacu and Peres [43]. □

We are now left to prove the other implication of Theorem 3.19: if a Dice Enterprise for $f : S \subset D^m \to \Delta^m$ has a fast simulation, then f must be real analytic.

Lemma 3.30. *Assume there exists a Dice Enterprise that has a fast simulation for $f : S \subset D^m \to (0,1)$. Then f is real analytic on S.*

Proof. We follow the idea of the proof of Proposition 20 by Nacu and Peres [43], here extended to the multivariate setting. Let $f^{(t)}(\boldsymbol{p})$ be the probability that the algorithm stops and outputs 1 exactly after t steps. Notice that this is slightly different than the definition we were using in the proof of Theorem 3.9, where $f^{(t)}(\boldsymbol{p})$ denoted the probability that the algorithm stops and outputs 1 within t steps. Define

$g^{(n)}(\boldsymbol{p}) := \sum_{t=1}^{n} f^{(t)}(\boldsymbol{p})$, so that the sequence $g^{(n)}(\boldsymbol{p})$ converges to $f(\boldsymbol{p})$ as $n \to \infty$. We shall prove that the sequence $\left\{g^{(n)}\right\}_{n\in\mathbb{N}}$ is Cauchy and conclude that its limit must be analytic by a standard theorem.

Let N be the number of $\boldsymbol{p}$-die rolls required by the algorithm to terminate. By hypothesis the Dice Enterprise has a fast simulation, thus:

$$\mathbb{P}_{\boldsymbol{p}}(N > n) \leq C\rho^n,$$

for some constants $C > 0, \rho < 1$. Fix any B such that $1 < B < 1/\rho$ and notice that since $f^{(n)}(\boldsymbol{p})$ is a polynomial, $f^{(n)}(\boldsymbol{z})$ is well defined for any complex $\boldsymbol{z} \in \mathbb{C}^m$. We first prove that there exists $\epsilon > 0$ such that for all $\boldsymbol{z} \in \mathbb{C}^m$ and $n > 0$, it holds:

$$|f^{(n)}(\boldsymbol{z})| \leq B^n f^{(n)}(\boldsymbol{p}) \qquad \text{if } |\boldsymbol{z} - \boldsymbol{p}| < \epsilon. \tag{3.8}$$

In this case, it is more convenient to work on the simplex space Δ^m rather than on the open disk D^m. Therefore, consider $\widetilde{\boldsymbol{p}} = (1 - \|\boldsymbol{p}\|_1, p_1, \ldots, p_m) \in \Delta^m$ and analogously $\widetilde{\boldsymbol{z}}$. We can equivalently rewrite $f^{(n)}(\boldsymbol{z})$ as:

$$f^{(n)}(\boldsymbol{z}) = f^{(n)}(\widetilde{\boldsymbol{z}}) = \sum_{\boldsymbol{k}\in\Lambda_n^m} a_{\boldsymbol{k}}^{(n)} \widetilde{\boldsymbol{z}}^{\boldsymbol{k}},$$

with $a_{\boldsymbol{k}}^{(n)} \geq 0, \forall \boldsymbol{k}, n$. Notice that if $|\boldsymbol{z} - \boldsymbol{p}| < \epsilon$, then $|\widetilde{\boldsymbol{z}}| < \widetilde{\boldsymbol{p}}_i + \epsilon, \forall i = 0, \ldots, m$. Choose ϵ so that $\widetilde{p}_i + \epsilon < B\widetilde{p}_i$ for all possible i, it follows:

$$\begin{aligned}\left|f^{(n)}(\boldsymbol{z})\right| = \left|\sum_{\boldsymbol{k}\in\Lambda_n^m} a_{\boldsymbol{k}}^{(n)} \widetilde{\boldsymbol{z}}^{\boldsymbol{k}}\right| &\leq \sum_{\boldsymbol{k}\in\Lambda_n^m} a_{\boldsymbol{k}}^{(n)} \left|\widetilde{\boldsymbol{z}}^{\boldsymbol{k}}\right| \leq \sum_{\boldsymbol{k}\in\Lambda_n^m} a_{\boldsymbol{k}}^{(n)} (\widetilde{\boldsymbol{p}} + \epsilon)^{\boldsymbol{k}} \\ &\leq \sum_{\boldsymbol{k}\in\Lambda_n^m} a_{\boldsymbol{k}}^{(n)} (B\widetilde{\boldsymbol{p}})^{\boldsymbol{k}} = \sum_{\boldsymbol{k}\in\Lambda_n^m} a_{\boldsymbol{k}}^{(n)} B^n \widetilde{\boldsymbol{p}}^{\boldsymbol{k}} = B^n f^{(n)}(\boldsymbol{p}),\end{aligned}$$

as desired. We now show that $\left\{g^{(n)}\right\}_n$ is indeed a Cauchy sequence. For any $m > n$

and $\boldsymbol{z}$ such that $|\boldsymbol{z}-\boldsymbol{p}|<\epsilon$, we use eq. 3.8 and the fact that $B\rho<1$ to show:

$$\begin{aligned}|g^m(\boldsymbol{p})-g^n(\boldsymbol{p})| &= \left|\sum_{t=n+1}^{m} f^{(t)}(\boldsymbol{p})\right| \leq \sum_{t=n+1}^{m}\left|f^{(t)}(\boldsymbol{p})\right| \leq \sum_{t=n+1}^{\infty} B^t f^{(t)}(\boldsymbol{p}) \\ &= \sum_{t=n+1}^{\infty} B^t \mathbb{P}(\text{algorithm stops and outputs 1 at the } t^{\text{th}} \text{ step}) \\ &\leq \sum_{t=n+1}^{\infty} B^t \mathbb{P}(\text{algorithm stops at the } t^{\text{th}} \text{ step}) = \sum_{t=n+1}^{\infty} B^t \mathbb{P}(N=t) \\ &\leq \sum_{t=n+1}^{\infty} B^t \mathbb{P}(N>t-1) \leq \sum_{t=n+1}^{\infty} B^t C\rho^{t-1} = \frac{BC(B\rho)^n}{1-B\rho}.\end{aligned}$$

Hence, for big n the difference $|g^m(\boldsymbol{p})-g^n(\boldsymbol{p})|$ can be made arbitrarily small, as desired. □

Although we have proved the result for functions on $(0,1)$, we can immediately conclude that for functions on Δ^v each f_i must be real analytic for $i=0,\ldots,v$. Therefore, by combining Lemmas 3.29 and 3.30 we have proved Theorem 3.19.

Example 3.31. Consider

$$f(p) := p\sqrt{2-p} = \sqrt{2}p - \frac{1}{\sqrt{2}}\sum_{n=2}^{\infty}\frac{2^{5-3n}}{n-1}\binom{2n-4}{n-2}p^n,$$

We can construct a Bernoulli Factory for $f(p)$ following Proposition 3.26. In particular, we separate the positive and negative coefficients and consider:

$$g(p) := \sqrt{2}p, \qquad h(p) := \frac{1}{\sqrt{2}}\sum_{n=2}^{\infty}\frac{2^{5-3n}}{n-1}\binom{2n-4}{n-2}p^n.$$

We can consider $t=1$, so that $f(t)=1$ and let:

$$M := \sqrt{2} + \frac{1}{\sqrt{2}}\sum_{n=2}^{\infty}\frac{2^{5-3n}}{n-1}\binom{2n-4}{n-2} = 2\sqrt{2}-1.$$

We now assume that $\bar{\epsilon},\underline{\epsilon}>0$ such that $\underline{\epsilon}\leq f(p)\leq 1-\bar{\epsilon}$ are known. Clearly tossing a $g(p)/M$-coin is easy, as $\sqrt{2}/M<1$. We can resort to Algorithm 14 to generate tosses of an $h(p)/M$-coin. Finally, we use Proposition 2.27 to toss an $f(p)/M$-coin and a linear Bernoulli Factory to toss the required $f(p)$-coin.

We set $\bar{\epsilon},\underline{\epsilon}$ so to have the best possible efficiency, i.e. $\bar{\epsilon}=1-f(p)$ and $\underline{\epsilon}=f(p)$. This is possible as we know the true value of p; in general such bounds will depend

on the problem specification.

p	0.1	0.5	0.9
$f(p)$	0.138	0.612	0.944
$\widehat{f}(p)$	$_{0.134}0.141_{\,0.147}$	$_{0.600}0.610_{\,0.619}$	$_{0.944}0.948_{\,0.952}$
$\widehat{\mathbb{E}}[N]$	252.165	124.730	533.419

Table 4: Implementation of a Bernoulli Factory for $f(p) = p\sqrt{2-p}$ and for different values of the true unknown probability p. The algorithm has been run 10,000 times to obtain tosses of the $f(p)$-coin and $\widehat{f}(p)$ is the sample average. Smaller number represent 95% confidence intervals computed via the Wilson score interval. $\widehat{\mathbb{E}}[N]$ is the empirical average number of tosses of the p-coin required.

3.5 Multiple independent coins

A closely related problem to the ones we have considered so far and that we briefly touched upon in Section 3.3, is the case where access to multiple independent coins is given. This case has already been considered in some applications [12, 16, 55] and we now encompass it in our theoretical framework. More formally, consider an unknown vector $\boldsymbol{q} = (q_0, \dots, q_{m-1}) \in (0,1)^m$ and that we are granted access to independent black-box mechanisms that output 1 with probability q_i and 0 otherwise. We denote the vector of coin probabilities by $\boldsymbol{q}$ rather than $\boldsymbol{p}$ to highlight that it does not represent dice probabilities, i.e. it may be $\|\boldsymbol{q}\| \neq 1$. The problem is then finding an algorithm that runs in finite time almost surely and returns rolls of an $f(\boldsymbol{q})$-die for a given function $f : (0,1)^m \to \Delta^v$. Consider the bijection $\phi : (0,1)^m \to H \subset \Delta^m$ defined as:

$$\begin{aligned} \phi(\boldsymbol{q}) &= \left(\frac{q_0}{m}, \dots, \frac{q_{m-1}}{m}, 1 - \frac{1}{m}\|\boldsymbol{q}\|_1\right), \\ \phi^{-1}(\boldsymbol{p}) &= (mp_0, \dots, mp_{m-1}), \end{aligned} \tag{3.9}$$

where $H := \{\boldsymbol{p} \in \Delta^m : p_i < 1/m,\ 0 \leq i < m\}$. We can use the same algorithm as in Lemma 3.8 to transform the given multiple coins to a die according to eq. (3.9). The theory we developed applies almost directly. For instance, we prove in the following corollary the analogous of Theorem 3.17 for the multiple coins case.

Corollary 3.32. *Given $(q_i)_{0\leq i<m}$-coins and $f(\boldsymbol{q}) : E \subset [0,1]^m \to \Delta^v$, an algorithm to roll an $f(\boldsymbol{q})$-die in finite time almost surely exists if and only if:*

- *Each $f_i(\boldsymbol{q})$ is continuous;*

- *There exists* $n_0 \in \mathbb{N}$ *such that each* $f_i(\boldsymbol{q})$ *satisfies* $f_i(\boldsymbol{q}) \geq \min(q_0, \dots, q_{m-1})^{n_0}$ *for all* $\boldsymbol{q} \in S$.

Proof. Consider $S := \phi(E)$, where ϕ is defined in equation (3.9), and let $g : S \subset \Delta^m \to \Delta^v$ such that $g(\boldsymbol{p}) = f\left(\phi^{-1}(\boldsymbol{p})\right), \forall \boldsymbol{p} \in S$. We show that g satisfies the assumptions of Theorem 3.9, so that there exists a Dice Enterprise for it. Clearly, each $g_i(\boldsymbol{p})$ is continuous as it is the composition of two continuous functions. To show that each $g_i(\boldsymbol{p})$ is polynomially bounded, note that:

$$\begin{aligned} g_i(\boldsymbol{p}) = f_i\left(\phi^{-1}(\boldsymbol{p})\right) &\geq \min(\phi_0^{-1}(\boldsymbol{p}), \dots, \phi_{m-1}^{-1}(\boldsymbol{p}))^{n_0} \\ &= \min(mp_0, \dots, mp_{m-1})^{n_0} \geq \min(p_0, \dots, p_{m-1})^{n_0}, \end{aligned}$$

as desired. Finally, given $(q_i)_{0 \leq i < m}$-coins we can construct via Lemma 3.8 a $\boldsymbol{p}$-die such that $\boldsymbol{p} \in S$ and $\boldsymbol{p} = \phi(\boldsymbol{q})$. By applying the Dice Enterprise for g on such die, we get the desired result by noticing that $g(\boldsymbol{p}) = f\left(\phi^{-1}(\phi(\boldsymbol{q}))\right) = f(\boldsymbol{q})$. □

3.5.1 Infinitely many coins and die faces

So far we have only considered the case of a finite number of coins or die faces. It is however interesting to consider the case where an infinite sequence of coins is given. We then study whether given an infinite sequence of $(p_i)_{i \in \mathbb{N}}$-coins where $\sum_{i=0}^{\infty} p_i = 1$, it is possible to design an algorithm that rolls a $\boldsymbol{p}$-die, i.e. such that the probability of rolling the i^{th} face is equal to p_i. This problem can also be posed differently as asking whether one can obtain samples from a discrete distribution on $\mathbb{N}$ where the probability mass function is unknown and for each $i \in \mathbb{N}$, only unbiased estimators $\widehat{p}_i \in [0, 1]$ of $p_i = \mathbb{P}(X = i)$ can be computed. Importantly, such estimators are mutually independent and therefore may not sum up to one, but they can be made arbitrarily close to p_i in probability. Ideally, we would like to prove an analogous of Lemma 3.8, that is whether we can construct a $\boldsymbol{p}$-die if and only if there exist algorithms that toss $(p_i)_{i \in \mathbb{N}}$-coins.

Clearly one implication follows easily: if an algorithm to roll such $\boldsymbol{p}$-die existed, we can toss a p_i-coin by just rolling the die and returning heads if the outcome is the i^{th} face, tails otherwise. Proving the other implication is significantly more challenging.

Remark 3.33. A natural first approach is trying to adapt the dice construction of Lemma 3.8. At each iteration of the algorithm we select a coin $g_K(\boldsymbol{p})$ to toss according to a fixed distribution $K \sim \boldsymbol{q}$, outputting K if it lands heads and restarting otherwise. The functions $g_k(\boldsymbol{p})$ need to be determined so that the algorithm produces

the desired output and it is possible to toss $(g_i(\boldsymbol{p}))_{i\in\mathbb{N}}$-coins. Let Y be the final output, then:

$$\mathbb{P}_{\boldsymbol{p}}(Y=j) = q_j g_j(\boldsymbol{p}) + \sum_{k=0}^{\infty} q_k(1-g_k(\boldsymbol{p}))\mathbb{P}_{\boldsymbol{p}}(Y=j),$$

and since $\sum_{k=0}^{\infty} q_k = 1$, one can see that it has to be for all $j \in \mathbb{N}$

$$p_j = \mathbb{P}_{\boldsymbol{p}}(Y=j) = \frac{q_j g_j(\boldsymbol{p})}{\sum_{k=0}^{\infty} q_k g_k(\boldsymbol{p})},$$

so that

$$g_j(\boldsymbol{p}) = \frac{p_j}{q_j}\sum_{k=0}^{\infty} q_k g_k(\boldsymbol{p}).$$

Therefore, the only valid choices are $g_j(\boldsymbol{p}) = Mp_j/q_j$ for any arbitrary constant $M > 0$. Nevertheless, these need to be valid probabilities for all possible $\boldsymbol{p}$, i.e. $g_j(\boldsymbol{p}) \in [0,1]$ for all possible $\boldsymbol{p}$. This condition cannot be met unless extra knowledge on $\boldsymbol{p}$ is available, so that $\boldsymbol{q}$ can be tuned accordingly. We conclude that there cannot be an algorithm that operates this way which works for all possible $\boldsymbol{p}$.

As a result of the above reasoning, we conjecture that it is impossible to construct an algorithm that rolls a $\boldsymbol{p}$-die in finite time almost surely unless extra information on $\boldsymbol{p}$ is available. An informal motivation being the following: if such algorithm existed, then it would be possible to toss a $\left(\sum_{i=0}^{K} p_i\right)$-coin for any choice of K by rolling the $\boldsymbol{p}$-die once and outputting heads if it results in a face smaller or equal than K, tails otherwise. Nevertheless, this is a linear Bernoulli Factory and we know that, in this case, extra knowledge is needed for an algorithm to exist (cf. Section 2.2).

We then propose two different algorithmic constructions that solve the problem of converting an infinite sequence of coins to a die with infinitely many faces which make use of additional assumptions on the vector of coin probabilities $\boldsymbol{p}$. The first one follows from the discussion of Remark 3.33, and assume that an upper bound $\delta_i \geq p_i$ is known for all i. In this case we can set $q_j = \delta_j/\|\boldsymbol{\delta}\|_1$ and $M = 1/(2\|\boldsymbol{\delta}\|_1)$, as detailed in the following proposition.

Proposition 3.34. *Given a sequence of $(p_i)_{i\in\mathbb{N}}$-coins where $\|\boldsymbol{p}\|_1 = 1$, an algorithm to roll a $\boldsymbol{p}$-die exists if a vector $\boldsymbol{\delta}$ such that $\|\boldsymbol{\delta}\|_1 < \infty$ and $f_i(\boldsymbol{p}) \leq \delta_i$ for all $i \in \mathbb{N}$ is known.*

Proof. First draw $K \sim \frac{\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|_1}$ and notice that

$$\sup_{i\in\mathbb{N}} \frac{\|\boldsymbol{\delta}\|_1 p_i}{\delta_i} \leq \|\boldsymbol{\delta}\|_1,$$

so that we can implement an acceptance-rejection algorithm: we output K with probability $\frac{1}{2\delta_K} p_K$ and restart otherwise. Let Y be the output of the algorithm; the probability that at each iteration the proposed point is accepted is given by:

$$\mathbb{P}(\text{accepted}) = \sum_{k=0}^{\infty} \frac{1}{2\delta_k} p_k \frac{\delta_k}{\|\boldsymbol{\delta}\|_1} = \frac{1}{2\|\boldsymbol{\delta}\|_1},$$

so that

$$\begin{aligned}\mathbb{P}(Y = i) &= \sum_{n=1}^{\infty} \mathbb{P}(\text{reject for } n-1 \text{ iterations and accept } i \text{ at the nth iteration}) \\ &= \sum_{n=1}^{\infty} \frac{\delta_i}{\|\boldsymbol{\delta}\|_1} \frac{1}{2\delta_i} p_i \left(1 - \frac{1}{2\|\boldsymbol{\delta}\|_1}\right)^{n-1} = p_i.\end{aligned}$$

To decide whether we accept K, we need to make use of a Bernoulli Factory sampling $X \sim \text{Bern}\left(\frac{1}{2\delta_K} p_K\right)$. If $2\delta_k \geq 1$ this is easy, otherwise we can resort to a linear Bernoulli Factory (cf. Section 2.2), since by hypothesis $\frac{1}{2\delta_K} p_K \leq \frac{1}{2}$. □

The other approach is sequential and at the i^{th} iteration we either output i or continue. To achieve this, at the 0^{th} iteration we can simply toss the p_0-coin and output 0 if it lands heads. Otherwise, we proceed to the next iteration and at the i^{th} iteration we output i with probability $p_i/(1 - \sum_{j=0}^{i-1} p_j)$. We then need to make use of the division Bernoulli Factory we have proposed in Section 2.2 and therefore assume that a lower bound $0 < \epsilon_i \leq p_i$ for all $i \in \mathbb{N}$ is known.

Proposition 3.35. *Given a sequence of $(p_i)_{i\in\mathbb{N}}$-coins where $\|\boldsymbol{p}\|_1 = 1$, an algorithm to roll a $\boldsymbol{p}$-die exists if a vector $\boldsymbol{\epsilon}$ such that $p_i \geq \epsilon_i > 0$ for all $i \in \mathbb{N}$ is known.*

Proof. We obtain a roll of the $\boldsymbol{p}$-die by tossing sequentially a $g_0(\boldsymbol{p})$-coin, $g_1(\boldsymbol{p})$-coin, and so on until we observe heads for the first time. We then output the index of the coin that landed heads as the rolled face. It is not hard to see that for this scheme to produce the required output, we must have

$$g_i(\boldsymbol{p}) = \frac{p_i}{1 - \sum_{j=0}^{i-1} p_j}.$$

We can toss a $\left(\sum_{j=0}^{i-1} p_j\right)$-coin (and then reverse the flip) via a linear Bernoulli Factory (cf. Section 2.2), which is implementable thanks to the following bound:

$$\sum_{j=0}^{i-1} p_j = 1 - \sum_{j=i}^{\infty} p_j \leq 1 - \sum_{j=i}^{\infty} \epsilon_j$$

Finally, we can resort to the division algorithm (cf. Algorithm 9) to get a flip of a $g_i(\boldsymbol{p})$-coin. □

Remark 3.36. Given a lower bound $\boldsymbol{\epsilon}$ such that $p_i \geq \epsilon_i > 0$ for all $i \in \mathbb{N}$ (thus satisfying the assumptions of Proposition 3.35), it is possible to construct an upper bound $\boldsymbol{\delta}$ by considering $\delta_i = \epsilon_i + 1 - \|\boldsymbol{\epsilon}\|_1$. However, such upper bound does not satisfy the assumptions of Propositions 3.34, in particular:

$$\|\boldsymbol{\delta}\|_1 = \|\boldsymbol{\epsilon}\|_1 + \sum_{i=0}^{\infty}(1 - \|\boldsymbol{\epsilon}\|_1),$$

which is infinite except in the case $\|\boldsymbol{\epsilon}\|_1 = 1$, corresponding to the case where $\boldsymbol{p}$ is known.

3.6 Discussion

This chapter formally introduced a multivariate extension of the Bernoulli Factory to categorical distributions and extended the fundamental available theoretical results to this novel case. We fully determined the class of functions for which a Dice Enterprise may be found and clarified the connection that exists between a Dice Enterprise for $f(\boldsymbol{p})$ and polynomial envelopes for the given function. Furthermore, we proposed efficient implementations for functions that admit a positive power series representation centred around the origin. Building on that, we proved that a Dice Enterprise admits a fast simulation if and only if it is real analytic on a set that is the Cartesian product of closed intervals. A natural open question, which we believe holds true, is whether our result extends to functions that are real analytic on any compact set. A possible way to prove such result would be to find a suitable mapping from such set to the probability simplex, so that a Dice Enterprise with a fast simulation could still be found.

Finally, we give initial insights into further extending our construction to any discrete distribution on a countably infinite support. In particular, we propose to analyse the problem of sampling from a discrete distribution $\boldsymbol{p}$ when only unbiased

estimators of each value p_i can be obtained to any arbitrary precision. The problem appears challenging as we assume such estimators to be mutually independent. By linking this problem to the theoretical framework we developed, we conjecture that unless extra knowledge on $\boldsymbol{p}$ is available, a solution cannot be found. A natural research path would then be to look into proving this result. Nevertheless, we give a solution to this problem when either upper or lower bounds of $\boldsymbol{p}$ are available.

4

A Dice Enterprise for Rational Functions via Perfect Sampling of Markov Chains

4.1 Introduction

This chapter introduces a constructive way to design a Dice Enterprise for rational functions based on perfect sampling from the stationary distribution of suitably constructed Markov chains. In Section 4.2 we introduce some fundamental preliminaries. In particular, we introduce ladders, a class of discrete probability distributions that are suitable as candidates for stationary distributions of Markov chains used in the sequel. We also summarise one of the main perfect simulation techniques, i.e. Coupling From the Past (CFTP), and present it in an algorithmic form for our case at hand. Section 4.3 develops the Dice Enterprise for rational functions $f : \Delta^m \to \Delta^v$. We first show how this problem can be rephrased as sampling from a ladder and construct a CFTP algorithm that performs it. We prove that our proposed construction is optimal in terms of Peskun's ordering. We then analyse the efficiency of the proposed algorithm in terms of the expected number of required rolls of the original die and notice that it is always finite under suitable assumptions. In the "coin to dice" scenario (of which the Bernoulli Factory is a special case), we notice that for log-concave ladders the expected number of tosses is linear in the degree of the ladder. We then prove that it is always possible to construct such log-concave distribution. Section 4.4 presents an `R` package that implements the developed method and explicative examples, validating the developed theory. In particular, we also show how a Dice Enterprise can be used to deal with m independent coins and reproduce examples taken from [12, 16, 24]. Proofs of all the results are presented in Section 4.6.

4.2 Preliminaries

Given a rational function $f : \Delta^m \to \Delta^v$ such that $f(\boldsymbol{p})$ is a valid discrete probability distribution for all $\boldsymbol{p} \in \Delta^m$, in Section 4.3 we will construct a new function $\pi : \Delta^m \to \Delta^k$, named ladder, such that $\pi(\boldsymbol{p})$ is also a valid discrete probability distribution and draws from $\pi(\boldsymbol{p})$ can be transformed into draws from $f(\boldsymbol{p})$ and vice-versa. To this end, consider pairs of distributions related by disaggregation defined as follows:

Definition 4.1 (Disaggregation)**.** Let $\boldsymbol{\mu} = (\mu_0, \ldots, \mu_k)$ and $\boldsymbol{\nu} = (\nu_0, \ldots, \nu_v)$ be probability distributions on Δ^k and Δ^v respectively with $v \leq k$. We say that $\boldsymbol{\mu}$ is a *disaggregation* of $\boldsymbol{\nu}$ if there exists a partition of $\{0, \ldots, k\}$ into $v+1$ sets $A_0, A_1, \ldots, A_v$ such that

$$\nu_i = \sum_{j \in A_i} \mu_j, \qquad \text{for all } i \in \{0, \ldots, v\}.$$

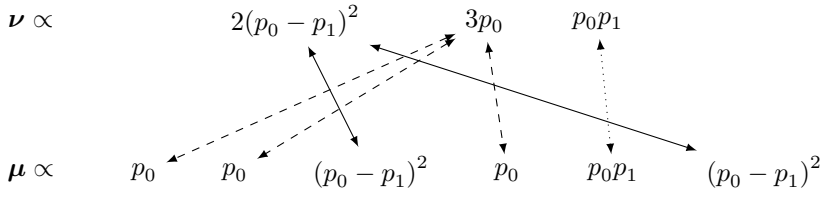
If $\boldsymbol{\mu}$ is a disaggregation of $\boldsymbol{\nu}$, then we shall equivalently say that $\boldsymbol{\nu}$ is an *aggregation* of $\boldsymbol{\mu}$.

If $\boldsymbol{\mu} = (\mu_0, \ldots, \mu_k)$ is a disaggregation of $\boldsymbol{\nu} = (\nu_0, \ldots, \nu_v)$ then, sampling from $\boldsymbol{\mu}$ is equivalent to sampling from $\boldsymbol{\nu}$ in the following sense:
Given $X \sim \boldsymbol{\mu}$, define Y as $Y := i$ if $X \in A_i$. Then $Y \sim \boldsymbol{\nu}$.
Given $X \sim \boldsymbol{\nu}$, define Y by letting

$$\mathbb{P}(Y = i) = \mathbb{I}(i \in A_X) \frac{\mu_i}{\sum_{j \in A_X} \mu_j}. \tag{4.1}$$

Then $Y \sim \boldsymbol{\mu}$.

Figure 1. Disaggregation. A sample from $\boldsymbol{\mu}$ is directly mapped to a sample from $\boldsymbol{\nu}$. A sample from $\boldsymbol{\nu}$ can be mapped to $\boldsymbol{\mu}$ proportionally.



Our proposed construction requires being able to draw exactly from the stationary distribution of a Markov chain, a deed that we achieve by perfect simulation

techniques. In particular, we will use Coupling From the Past (CFTP) [52] a review of which is presented in the following section.

4.2.1 Coupling from the past

Perfect sampling is a well developed approach [26] to devise specialised algorithms, necessarily with random running time, that will output a single draw exactly from the stationary distribution of a Markov chain, rather than from its approximation. Coupling From the Past (CFTP) [52] is a pioneering technique in this field and illustrative for our purposes. The idea behind the method relies on starting the chain at time $-\infty$, so that at present time one would have a sample from the stationary distribution. This may not seem practical, but as pointed out in [52], one can make use of coupled chains to decide when to stop tracking the chain back in time. In practice, it is convenient to introduce an update function for the chain. Given a state i and a source of randomness, the update function returns the state of the chain at the next step. Such source of randomness is commonly represented by a single draw from a uniform random variable U, as one could then consider its binary representation to have an arbitrary number of uniform random variables and transform them, at least in principle, via inversion sampling. However, for our specific application, it is natural to consider the given die a source of randomness. As discussed in Section 1.4, we can resort to the die to generate uniform random variables. However, for better clarity, we consider having access to both the die and a uniform random variable as source of randomness for the update function, defined as follows:

Definition 4.2 (Update function)**.** Let $(X_t)_{t\in\mathbb{N}}$ be a Markov chain on $\Omega = \{0, 1, \dots, k\}$. Assume $\boldsymbol{p} \in \Delta^m$ and let $B \sim \boldsymbol{p}$ and $U \sim \text{Unif}(0, 1)$. A function

$$\phi : \Omega \times \{0, 1, \dots, m\} \times [0, 1] \to \Omega$$

is an *update function* for the Markov chain $(X_t)_{t\in\mathbb{N}}$ if

$$\mathbb{P}(X_{t+1} = j | X_t = i) = \mathbb{P}(\phi(i, B, U) = j), \qquad \forall i, j \in \Omega.$$

We will write $\phi_t(x, \boldsymbol{B}, \boldsymbol{U}) = \phi(\phi(\dots(\phi(x, B_1, U_1), B_2, U_2), \dots), B_t, U_t)$ to indicate the state of the chain after t steps when starting from x.

Given an update function ϕ, CFTP is implementable via Algorithm 15. Under mild assumptions, namely that there is a positive probability of termination, Algorithm 15 produces samples from the stationary distribution of the Markov chain

[26, 52].

Algorithm 15 Coupling From the Past

Input: an update function ϕ for a Markov chain $(X_t)_{t\in\mathbb{N}}$ on $\Omega = \{0, \dots, k\}$ with unique stationary distribution π; a black box to sample from $\boldsymbol{p} \in \Delta^m$.

Output: A sample from π.

1: Set $T \leftarrow 1$
2: **for** $i = 0, \dots, k$ **do** $X_0^{(i)} \leftarrow i$ **end for**
3: **repeat**
4: Sample independently $B_{-T} \sim \boldsymbol{p}$ and $U_{-T} \sim \mathrm{Unif}(0, 1)$
5: **for** $i = 0, \dots, k-1$ **do** $X_0^{(i)} \leftarrow \phi_T(i, (B_{-T}, \dots, B_{-1}), (U_{-T}, \dots, U_{-1}))$ **end for**
6: Set $T \leftarrow T + 1$
7: **until** $X_0^{(0)} = X_0^{(1)} = \ldots = X_0^{(k)}$
8: **Output** $X_0^{(0)}$

Notice that CFTP needs to keep track of the trajectories of k coupled chains. If k is large implementing the algorithm may become infeasible. A more efficient version of CFTP can be designed for monotonic Markov chains [52]. In particular, assume that the state space Ω of the Markov chain $(X_t)_{t\in\mathbb{N}}$ admits a partial order $\preceq$, and there exist the maximum and the minimum states, say 0 and k, respectively; i.e., $\forall j \in \Omega$, $j \preceq k$ and $0 \preceq j$. The monotonic update function is defined as follows.

Definition 4.3 (Monotonic update function)**.** An update function ϕ as in Definition 4.2 is *monotonic* if for all $B \in \{0, 1, \dots, m\}$ and $U \in [0, 1]$

$$i \preceq j \quad \Longrightarrow \quad \phi(i, B, U) \preceq \phi(j, B, U). \tag{4.2}$$

In the monotonic case it is enough to track coalescence of just two chains, started from the minimum and the maximum state. Algorithm 16 presents CFTP with monotonic update function ϕ.

Algorithm 16 Monotonic Coupling From the Past

Input: a monotonic update function ϕ for a Markov chain $(X_t)_{t\in\mathbb{N}}$ on $\Omega = \{0,\dots,k\}$ with minimum and maximum states, 0 and k respectively and with unique stationary distribution π; a black box to sample from $\boldsymbol{p} \in \Delta^m$.

Output: A sample from π.

1: Set $T \leftarrow 1$, $X_0 \leftarrow 0$ and $Y_0 \leftarrow k$
2: **repeat**
3: Sample independently $B_{-T} \sim \boldsymbol{p}$ and $U_{-T} \sim \mathrm{Unif}(0,1)$
4: Set $X_0 \leftarrow \phi_T(0, (B_{-T},\dots,B_{-1}), (U_{-T},\dots,U_{-1}))$
5: Set $Y_0 \leftarrow \phi_T(k, (B_{-T},\dots,B_{-1}), (U_{-T},\dots,U_{-1}))$
6: Set $T \leftarrow T+1$
7: **until** $X_0 = Y_0$
8: **Output** X_0

Remark 4.4. Commonly, monotonic CFTP is implemented so that T is doubled at each iteration (replacing line 6 of the algorithm with $T \leftarrow 2T$), so to optimise the number of times ϕ_T needs to be evaluated. To see why, let $T_\star$ be the coalesce time, i.e. the first time such that $X_0 = Y_0$ in Algorithm 16. Then, increasing T by one at each iteration will result in $(T_\star^2 + T_\star)$ simulation steps and $T_\star$ rolls of the $\boldsymbol{p}$-die. On the other hand, if T is doubled at each iteration, the algorithm will require at most $8T_\star$ simulation steps (cf. Section 5.2 of Propp and Wilson [52]), but $2^{\lceil \log_2(T_\star)\rceil}$ rolls of the $\boldsymbol{p}$-die. There is therefore a trade-off which depends also on the computational effort that comes with rolling the given die. In the following, we will just consider $T \leftarrow T+1$ as presented in Algorithm 16 so to minimise the required number of dice rolls.

4.2.2 Ladder over $\mathbb{R}$

We now introduce a class of probability distributions $\pi : \Delta^m \to \Delta^k$ that will be of main interest in the remainder of the chapter. Recall that the 1-norm of a vector $\boldsymbol{a} = (a_0,\dots,a_n)$ is $\|\boldsymbol{a}\|_1 = \sum_{j=0}^n |a_j|$.

Definition 4.5 (Multivariate ladder)**.** For every $\boldsymbol{p} = (p_0,\dots,p_m) \in \Delta^m$ let $\pi(\boldsymbol{p}) = (\pi_0(\boldsymbol{p}),\dots,\pi_k(\boldsymbol{p}))$ be a probability distribution on $\Omega = \{0,\dots,k\}$. We say that $\pi(\boldsymbol{p})$ is a *multivariate ladder* over $\mathbb{R}$ if every π_i is of the form

$$\pi_i(\boldsymbol{p}) = R_i \frac{\prod_{j=0}^m p_j^{n_{i,j}}}{C(\boldsymbol{p})}, \tag{4.3}$$

where

- $C(\boldsymbol{p})$ is a polynomial with real coefficients that does not admit roots in $\bar{\Delta}^m$;
- $\forall i, j$, R_i is a strictly positive real constant and $n_{i,j} \in \mathbb{N}_{\geq 0}$;
- Denote $\boldsymbol{n}_i = (n_{i,0}, n_{i,1}, \ldots, n_{i,m})$. There exists an integer d such that $\|\boldsymbol{n}_i\|_1 = d$ for all i. We will refer to $\boldsymbol{n}_i$ as the degree of $\pi_i(\boldsymbol{p})$ and to d as the degree of $\pi(\boldsymbol{p})$.

Moreover, we say that $\pi(\boldsymbol{p})$ is a *connected ladder* if

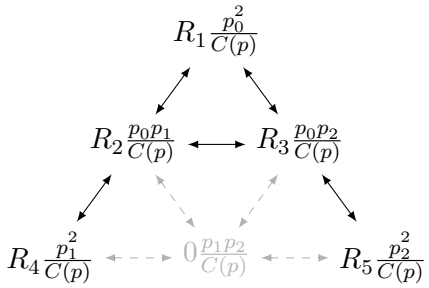
- Each $i, j \in \Omega$ are connected, meaning that there exists a sequence of states $(i = s_1, s_2, ..., s_t = j)$, such that $\left\|\boldsymbol{n}_{s_h} - \boldsymbol{n}_{s_{(h-1)}}\right\|_1 \leq 2$ for all $h \in \{2, \ldots, t\}$.

Finally, we say that $\pi(\boldsymbol{p})$ is a *fine ladder* if

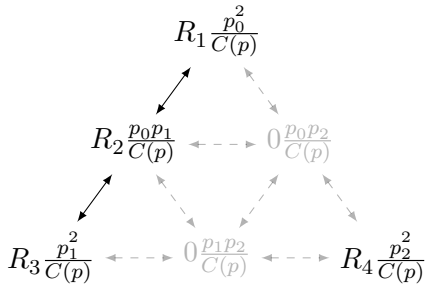
- $\boldsymbol{n}_i = \boldsymbol{n}_j$ implies $i = j$.

Figure 2 gives a graphical representation of a multivariate ladder and motivates the following concept of neighbourhood.

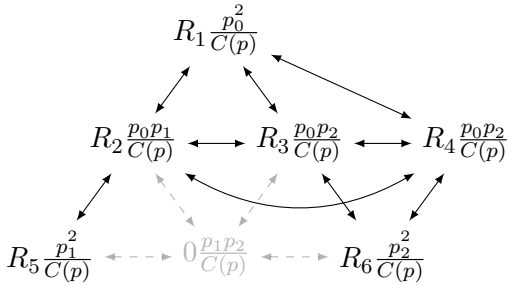
Definition 4.6 (Neighbourhoods on ladders)**.** On a multivariate ladder $\pi : \Delta^m \to \Delta^k$ define the *neighbourhood of* $i \in \Omega$ as $\mathcal{N}(i) = \{j \in \Omega \setminus \{i\} : \left\|\boldsymbol{n}_i - \boldsymbol{n}_j\right\|_1 \leq 2\}$. Note that for connected ladders $\mathcal{N}(i)$ must have at least one element for each i (in the non-trivial case of $k > 1$).



(A) Fine and connected ladder $\pi : \Delta^2 \to \Delta^4$



(B) Fine, but not connected ladder $\pi : \Delta^2 \to \Delta^3$



(C) Connected, but not fine ladder $\pi : \Delta^2 \to \Delta^5$

FIGURE 2. Multivariate ladders over $\mathbb{R}$. Edges represent connected states.

Operations on ladders

We now introduce three operations on ladders of which we will make extensive use: *increasing the degree* of a ladder, *thinning* and *augmenting* a ladder.

Definition 4.7 (Increasing the degree)**.** Let $\pi : \Delta^m \to \Delta^k$ be a multivariate ladder of degree d. *Increasing the degree of* π yields a new ladder $\pi' : \Delta^m \to \Delta^{(k+1)(m+1)-1}$ of degree $(d+1)$ with probabilities $\pi'_l(\boldsymbol{p})$ on $\Omega' = \{0, \ldots, (k+1)(m+1)-1\}$ of the form $\pi'_l(\boldsymbol{p}) := \pi_i(\boldsymbol{p})p_j$, where $i \in \{0, \ldots, k\}$ and $j \in \{0, \ldots, m\}$ are the unique solution of $l = i(m+1) + j$.

Increasing the degree corresponds to multiplying each state by $p_0, \ldots, p_m$ and the resulting ladder $\pi'(\boldsymbol{p})$ is a disaggregation of $\pi(\boldsymbol{p})$. Indeed, let $A_0, \ldots, A_k$ be

$$A_i := \{i(m+1), \ldots, i(m+1)+m\}.$$

Then Definition 4.1 is satisfied, since

$$\sum_{a\in A_i} \pi'_a(\boldsymbol{p}) = \sum_{j=0}^{m} \pi_i(\boldsymbol{p})p_j = \pi_i(\boldsymbol{p}) \sum_{j=0}^{m} p_j = \pi_i(\boldsymbol{p}).$$

Definition 4.8 (Thinning)**.** Let $\pi : \Delta^m \to \Delta^k$ be a multivariate ladder of degree d. *Thinning* π yields a fine ladder π' by joining all the states of π with the same monomial. Thus $\pi' : \Delta^m \to \Delta^w$ where $k \geq w := |\{\boldsymbol{n}_0, \ldots, \boldsymbol{n}_k\}|$, and by (4.3) the probabilities $\pi'_l(\boldsymbol{p})$ on $\Omega' = \{0, \ldots, w\}$ are of the form $\pi'_l(\boldsymbol{p}) := \frac{R'_l \boldsymbol{p}^{\boldsymbol{n}'_l}}{C(\boldsymbol{p})}$ where $R'_l = \sum_{i:\boldsymbol{n}_i=\boldsymbol{n}'_l} R_i$.

Clearly, $\pi(\boldsymbol{p})$ is a disaggregation of the resulting $\pi'(\boldsymbol{p})$. Moreover, if π is a connected ladder, then so is π'.

Increasing the degree will typically not result in a fine ladder as it produces "redundant" states, however thinning can be applied subsequently. We will refer to increasing the degree of the ladder first and thinning it afterwards, as *augmenting* the ladder.

Definition 4.9 (Augmenting)**.** Let $\pi : \Delta^m \to \Delta^k$ be a multivariate ladder of degree d. The augmented ladder $\pi' : \Delta^m \to \Delta^w$, where $w < \min\left\{(k+1)(m+1), \binom{d+m+1}{m}\right\}$, is obtained by first increasing the degree of π and then thinning it.

The fact that $w < \binom{d+m+1}{m}$ in the augmented ladder of degree $d+1$, follows by noticing that there are at most $\binom{d+m-1}{m-1}$ homogeneous monomials of degree d in m variables. Importantly, sampling from π and its augmented ladder π' is equivalent, since it is enough to transform the sample in line with the disaggregation and aggregation steps applied. Finally we make the following important remark that connects the operation of augmenting a ladder to that of convolution of random variables.

Remark 4.10. Let $m = 1$, π be a fine ladder and assume $\boldsymbol{n}_i$'s are ordered lexicographically. Moreover, let $W \sim \mathrm{Ber}(p)$ be independent of $X \sim \pi(p)$ and Y be the convolution of the two, that is $Y = X + W$. Then, $Y \sim \pi'(p)$, where π' is the augmented π.

Notice that given a multivariate ladder $\pi : \Delta^m \to \Delta^k$, augmenting it enough times yields a fine and connected ladder.

Proposition 4.11. *Let $\pi : \Delta^m \to \Delta^k$ be a multivariate ladder of degree d. Augment π d times to construct $\pi' : \Delta^m \to \Delta^w$, where $w < \min\left\{(k+1)(m+1)^d, \binom{2d+m}{m}\right\}$. Then π' is a fine and connected ladder and sampling from π' is equivalent to sampling from π.*

In practice, it may be enough to augment the ladder π less than d times to produce a fine and connected ladder π'.

Univariate fine and connected ladder over $\mathbb{R}$

Consider the special case of fine and connected ladders where $m = 1$, which will be of particular interest for the monotone CFTP implementation. We shall call such ladders univariate and denote $p_0 = 1 - p$ and $p_1 = p$. The condition that $C(p)$ has no roots in $[0, 1]$ implies $d = k$ and the probabilities take the form of

$$\pi_i(p) = R_i \frac{p^i (1-p)^{k-i}}{C(p)}, \qquad i = 0, \ldots, k. \tag{4.4}$$

This case corresponds to having access to a p-coin and simulating a $(k+1)$-sided die, so that the classic Bernoulli Factory setting falls in this scenario.

4.3 A dice enterprise for rational functions

We now tackle the problem of designing an algorithm that given a die where the probability $\boldsymbol{p} \in \Delta^m$ of rolling each face is unknown, produces rolls of a die (possibly having a different number of faces v) where the probability associated to each face is $f(\boldsymbol{p})$, i.e. given by a rational function $f : \Delta^m \to \Delta^v$. We first show how to decompose a rational function f into a fine and connected ladder $\pi : \Delta^m \to \Delta^k$ such that sampling from $f(\boldsymbol{p})$ is equivalent to sampling from $\pi(\boldsymbol{p})$. Then, we detail how to construct a Markov chain that admits such $\pi(\boldsymbol{p})$ as its stationary distribution. Finally, we apply CFTP perfect sampling technique to get a sample exactly from $\pi(\boldsymbol{p})$. The algorithm produces exact draws from $\pi(\boldsymbol{p})$ for any m, but for the case $m = 1$ the CFTP has a monotonic implementation with improved efficiency.

4.3.1 Construction of π

Theorem 4.12 below shows that for any rational function $f : \Delta^m \to \Delta^v$, a fine and connected ladder $\pi : \Delta^m \to \Delta^k$ can be constructed, such that sampling from f is equivalent to sampling from π. The construction is explicit and among others, builds on the ideas of Mossel et al. [42].

Roughly speaking the key steps of our proposed method are the following:

1: Let $f(\boldsymbol{p}) = (f_0(\boldsymbol{p}), \ldots, f_v(\boldsymbol{p})) = \left(\frac{D_0(\boldsymbol{p})}{E_0(\boldsymbol{p})}, \ldots, \frac{D_v(\boldsymbol{p})}{E_v(\boldsymbol{p})}\right)$ be a given rational function where $D_i(\boldsymbol{p})$ and $E_i(\boldsymbol{p})$ are positive and relatively prime polynomials.

2: Apply Lemma 4.33 (presented in Section 4.6) to each $f_i(\boldsymbol{p})$, so that $f(\boldsymbol{p}) = \left(\frac{d_0(\boldsymbol{p})}{e_0(\boldsymbol{p})}, \ldots, \frac{d_v(\boldsymbol{p})}{e_v(\boldsymbol{p})}\right)$ and each $d_i(\boldsymbol{p})$ and $e_i(\boldsymbol{p})$ is an homogeneous polynomial with positive coefficients.

3: Rewrite $f(\boldsymbol{p})$ using a common denominator, so that $f(\boldsymbol{p}) = \frac{1}{C(\boldsymbol{p})}(G_0(\boldsymbol{p}), \ldots, G_v(\boldsymbol{p}))$.

4: Rewrite each polynomial $G_i(\boldsymbol{p})$ as a homogeneous polynomial of degree d.

5: Using Proposition 4.11, construct a fine and connected ladder $\pi(\boldsymbol{p})$ sampling from which is equivalent to sampling from $f(\boldsymbol{p})$.

Detailed construction can be found in the proof of the following theorem and is also illustrated in Example 4.28.

Theorem 4.12. *Let $f : \Delta^m \to \Delta^v$ be a probability distribution such that every $f_i(\boldsymbol{p})$ is a rational function with real coefficients. Then, one can explicitly construct a fine and connected ladder $\pi : \Delta^m \to \Delta^k$ such that sampling from π is equivalent to sampling from f.*

Notice that we assume that $f(\boldsymbol{p}) \in \Delta^v$ for every $\boldsymbol{p} \in \Delta^m$. This rules out functions such as $f(p) = \min(2p, 1)$ - as expected since a Bernoulli Factory for such function cannot be constructed (cf. discussion in Section 1.1).

4.3.2 Construction of the Markov chain

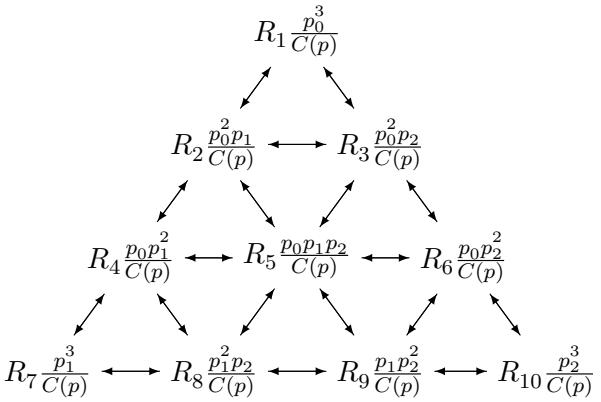
Let $\pi : \Delta^m \to \Delta^k$ be a fine and connected ladder. We now consider the problem of designing a Markov chain that admits it as its stationary distribution. The main idea behind the construction is depicted in Figure 3: a roll of the die determines the possible directions for the next move. We then draw a Uniform r.v. and decide whether the chain stays still or moves to a specific state. We can then write the off diagonal entries of the transition matrix P of the chain as an entrywise product of V and W, i.e.

$$P_{i,j} = \begin{cases} V_{i,j} \cdot W_{i,j} & \text{if } i \neq j \\ 1 - \sum_{h \neq i} P_{i,h} & \text{if } i = j \end{cases} \tag{4.5}$$

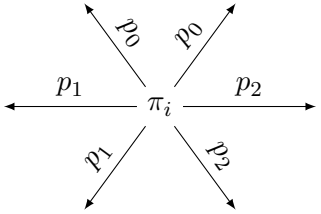
where V is a matrix of real numbers in $[0, 1]$ and W is a matrix where the off diagonal elements are either null or equal to p_b for some $b \in \{0, \ldots, m\}$.

It is convenient to introduce the following definition of neighbourhood that, once the $(m+1)$-sided die has been rolled, details where the chain may move.

Definition 4.13. Let $\pi : \Delta^m \to \Delta^k$ be a fine and connected multivariate ladder. Let $b \in \{0, \ldots, m\}$ and let $\boldsymbol{e}_b \in \bar{\Delta}^m$ be the b^{th} standard unit vector. For each



(A) A connected and fine ladder $\pi : \Delta^2 \to \Delta^9$



(B) Possible moves given a roll of the die

Figure 3. Markov chain structure for a multivariate ladder

$i \in \Omega = \{0, \ldots, k\}$ define the neighbourhood of i in the direction of b as

$$\mathcal{N}_b(i) = \{j \in \mathcal{N}(i) : (\boldsymbol{n}_j)_b = (\boldsymbol{n}_i)_b + 1\}, \tag{4.6}$$

where $(\boldsymbol{n}_j)_b$ represents the element at the b^{th} position of vector $\boldsymbol{n}_j$ and $\mathcal{N}(i)$ is the neighbourhood of i (cf. Definition 4.6). We will also denote

$$\mathcal{S}_b(i) = \sum_{j \in \mathcal{N}_b(i)} R_j, \tag{4.7}$$

where recall that R_j denotes the ladder's coefficients (cf. Definition 4.5).

Unlike $\mathcal{N}(i)$ (cf. Definition 4.6), $\mathcal{N}_b(i)$ may be empty and

$$\mathcal{N}(i) = \bigcup_{b \in \{0,\ldots,m\}} \mathcal{N}_b(i).$$

We can now set the elements of the matrix W in (4.5) as

$$W_{i,j} = \begin{cases} p_b & \text{if } j \in \mathcal{N}_b(i) \\ 0 & \text{if } j \notin \mathcal{N}(i). \end{cases} \tag{4.8}$$

We are left to define the matrix V in eq. (4.5). In principle, to speed up the convergence of CFTP it is good practice to reduce the mixing time of the chain [52]. A related and more operational criterion is that of Peskun ordering [49]:

Definition 4.14. Given two reversible Markov chains with the same stationary

distribution π and with transition matrices P and Q, we say that Q dominates P in Peskun sense, and write $Q \succeq_P P$, if each of the off diagonal elements of Q are greater or equal to the corresponding elements of P.

If $Q \succeq_P P$ then, for any π integrable target function f, using Q results in a smaller asymptotic variance in the Markov chain CLT than using P. Moreover, for positive operators, if $Q \succeq_P P$ then the Q-chain converges in total variation more rapidly towards the stationary distribution [40]. Consequently, Peskun ordering represents a valuable tool to assess our choice of the transition matrix.

The requirement of having a reversible Markov chain leads to a natural choice for the off-diagonal entries of the matrix V:

$$V_{i,j} = \frac{R_j}{\mathcal{S}_b(i) \vee \mathcal{S}_c(j)} \qquad \text{if } j \in \mathcal{N}_b(i) \text{ and } i \in \mathcal{N}_c(j). \tag{4.9}$$

It is easy to check that this choice produces a transition matrix P that satisfies the detailed balance condition and that $\pi(\boldsymbol{p})$ is the stationary distribution of the chain. However, this choice may not be optimal in Peskun's ordering. Therefore, we propose a different construction and define the matrix V of eq. (4.5) iteratively. First, we select the state $i \in \Omega$ and the roll $b \in \{0, \dots, m\}$ that maximises $\mathcal{S}_b(i)$; i.e. we identify the pair of state and die face from which it is easiest for the chain to move away from. In particular, for each $j \in \mathcal{N}_b(i)$ we assign $V_{i,j} = \frac{R_j}{\mathcal{S}_b(i)}$. Since the construction shall yield a reversible Markov chain, we also set $V_{j,i} = \frac{R_i}{\mathcal{S}_b(i)}$ (notice that the denominator is purposely set equal to $\mathcal{S}_b(i)$ so that detailed balance condition is satisfied). We then proceed analogously, now identifying a new pair of state and die face for which the probability of moving out from such state (going in the direction of the given face) is maximised, taking into account the entries of V that have already been fixed. The detailed procedure is described in Algorithm 17. We prove in Proposition 4.15 that since at each step we ensure that the detailed balance condition holds, this leads to a valid reversible chain and $\pi(\boldsymbol{p})$ is its unique stationary distribution. Importantly, the so-constructed transition matrix P is optimal in Peskun ordering. We prove this in Proposition 4.16, the key point being that since at each step we select the pair of state and die face that maximises the off-diagonal elements of V, any other choice would produce a matrix P that is no longer a valid stochastic matrix for all $\boldsymbol{p} \in \Delta^m$.

Proposition 4.15. *Let $\pi : \Delta^m \to \Delta^k$ be a fine and connected ladder. Consider a discrete-time Markov chain $(X_t)_{t\in\mathbb{N}}$ on $\Omega = \{0, \dots, k\}$. Let P as in (4.5) be the transition matrix of the chain, where W is as in (4.8) and V is the matrix output*

Algorithm 17 Construction of the Markov chain transition matrix

Input: A multivariate ladder $\pi : \Delta^m \to \Delta^k$ on $\Omega = \{0, \dots, k\}$
Output: The matrix V composing the transition kernel in equation (4.5).

Initialisation step

Initialise V as a $(k+1) \times (k+1)$ null matrix
For all $i \in \Omega$ and $b \in \{0, \dots, m\}$ set $\mathcal{N}_b(i)$ as in eq. (4.6), $\mathcal{S}_b(i)$ as in eq. (4.7), $\mathcal{W}_b(i) \leftarrow 0$
Main loop
repeat
    Set $b, i \leftarrow \arg\max_{b,i} \mathcal{S}_b(i)$
    for each $j \in \mathcal{N}_b(i)$ **do**
Assign maximum probability of moving from state i having rolled b
        $V_{i,j} \leftarrow R_j / S_b(i)$, $\mathcal{N}_b(i) \leftarrow \mathcal{N}_b(i) \setminus \{j\}$, $\mathcal{W}_b(i) \leftarrow \mathcal{W}_b(i) + R_j / S_b(i)$
Assign values for the reverse move
        Set c such that $i \in \mathcal{N}_c(j)$
        $V_{j,i} \leftarrow R_i / S_b(i)$, $\mathcal{N}_c(j) \leftarrow \mathcal{N}_c(j) \setminus \{i\}$, $\mathcal{W}_c(j) \leftarrow \mathcal{W}_c(j) + R_i / S_b(i)$
Update $S_c(j)$ to take into consideration the new value of $V_{j,i}$
        $S_c(j) \leftarrow \left(\sum_{h \in \mathcal{N}_c(j)} R_h \right) (1 - \mathcal{W}_c(j))^{-1}$
    end for
Update $\mathcal{S}_b(i)$
    $\mathcal{S}_b(i) \leftarrow 0$
until $\mathcal{N}_b(i) = \emptyset, \forall i \in \Omega, b \in \{0, \dots, m\}$

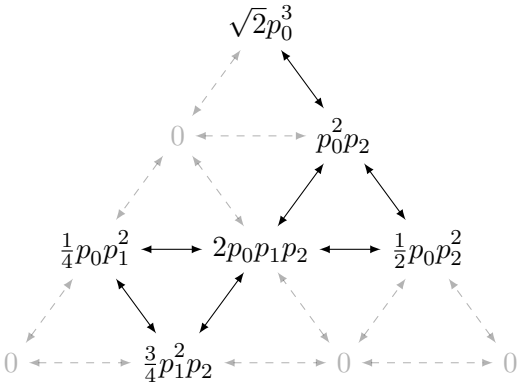
by Algorithm 17. Then, $(X_t)_{t \in \mathbb{N}}$ is a time-reversible Markov chain that admits $\pi(\boldsymbol{p})$ as its unique stationary distribution.

Proposition 4.16. *Let $\pi : \Delta^m \to \Delta^k$ be a fine and connected ladder. Consider the Markov chain defined in Proposition 4.15. Then, there does not exist a reversible Markov chain with the same adjacency structure and stationary distribution that dominates it in the Peskun sense.*

Example 4.17. Consider the multivariate ladder

$$\pi(p_0, p_1, p_2) \propto \left(\underbrace{\sqrt{2} p_0^3}_{\pi_0}, \underbrace{p_0^2 p_2}_{\pi_1}, \underbrace{\frac{1}{4} p_0 p_1^2}_{\pi_2}, \underbrace{2 p_0 p_1 p_2}_{\pi_3}, \underbrace{\frac{1}{2} p_0 p_2^2}_{\pi_4}, \underbrace{\frac{3}{4} p_1^2 p_2}_{\pi_5} \right).$$

We can graphically represent the ladder as



The neighbourhoods of each state are

$$\begin{aligned}
&\mathcal{N}(0) = \{1\}, && \mathcal{N}(1) = \{0,3,4\}, && \mathcal{N}(2) = \{3,5\},\\
&\mathcal{N}(3) = \{1,2,4,5\}, && \mathcal{N}(4) = \{1,3\}, && \mathcal{N}(5) = \{2,3\}.
\end{aligned}$$

and given how we defined $\mathcal{N}_b(i)$ we have

$$\begin{aligned}
&\mathcal{N}_0(0) = \emptyset, && \mathcal{N}_1(0) = \emptyset, && \mathcal{N}_2(0) = \{1\},\\
&\mathcal{N}_0(1) = \{0\}, && \mathcal{N}_1(1) = \{3\}, && \mathcal{N}_2(1) = \{4\},\\
&\mathcal{N}_0(2) = \emptyset, && \mathcal{N}_1(2) = \emptyset, && \mathcal{N}_2(2) = \{3,5\},\\
&\mathcal{N}_0(3) = \{1\}, && \mathcal{N}_1(3) = \{2,5\}, && \mathcal{N}_2(3) = \{4\},\\
&\mathcal{N}_0(4) = \{1\}, && \mathcal{N}_1(4) = \{3\}, && \mathcal{N}_2(4) = \emptyset,\\
&\mathcal{N}_0(5) = \{2,3\}, && \mathcal{N}_1(5) = \emptyset, && \mathcal{N}_2(5) = \emptyset.
\end{aligned}$$

The transition matrix obtained through Algorithm 17 is then equal to

$$P = \begin{pmatrix}
\cdot & \frac{1}{\sqrt{2}}p_2 & 0 & 0 & 0 & 0\\
p_0 & \cdot & 0 & p_1 & \frac{1}{2}p_2 & 0\\
0 & 0 & \cdot & \frac{8}{11}p_2 & 0 & \frac{3}{11}p_2\\
0 & \frac{1}{2}p_0 & \frac{1}{11}p_1 & \cdot & \frac{15}{44}p_2 & \frac{1}{3}p_1\\
0 & p_0 & 0 & p_1 & \cdot & 0\\
0 & 0 & \frac{1}{11}p_0 & \frac{10}{11}p_0 & 0 & \cdot
\end{pmatrix}$$

where $\cdot$ represents the required quantity so that the rows sum up to 1.

4.3.3 Perfect sampling

We now introduce an update function for the Markov chain defined in Proposition 4.15 so that CFTP is implementable. Given the current state, a roll of the die B, and a draw from a uniform random variable U, the update function details where the chain moves next (its formal definition can be found in Section 4.2.1). This motivates the choice of defining the transition matrix as the element-wise product of two matrices (cf. equation (4.5)): if the chain is in state i, we attempt to move to any state j such that $W_{i,j} = p_B$ and reach a final decision by comparing U and the values of $V_{i,j}$. We are then able to draw samples from a multivariate fine and connected ladder and thus solve the original problem via Theorem 4.12. For the general case of a die with more than 2 faces, the update function defined in the following proposition is not necessarily monotonic. However, in the Bernoulli Factory setting of $m = 1$, we can define a monotonic update function for the Markov chain as shown in Corollary 4.19. Notice that even when a monotonic construction is not available, CFTP can still be used in practice. As numerical examples demonstrate (cf. Examples 4.29, 4.31), if the degree of the polynomials and the numbers of faces of the given die are not too large, running times are not prohibitive.

Proposition 4.18. *Given a fine and connected ladder $\pi : \Delta^m \to \Delta^k$, consider the Markov chain $(X_t)_{t\in\mathbb{N}}$ with transition matrix P of the form (4.5) and defined in Proposition 4.15. Let $B \sim \boldsymbol{p}$ and $U \sim Unif(0,1)$ be independent random variables. Given $i \in \Omega$ denote the elements of $\mathcal{N}_B(i)$ as $\mathcal{N}_B(i) = \{j_0, \dots, j_w\}$. Define the function $\phi : \{0,\dots,k\} \times \{0,\dots,m\} \times [0,1] \to \{0,\dots,k\}$ as*

$$\phi(i,B,U) = \begin{cases} j_0 & \text{if } U \le V_{i,j_0}, \\ j_1 & \text{if } V_{i,j_0} < U \le V_{i,j_0} + V_{i,j_1}, \\ \quad\dots & \\ j_l & \text{if } \sum_{h=0}^{l-1} V_{i,j_h} < U \le \sum_{h=0}^{l} V_{i,j_h}, \\ \quad\dots & \\ i & \text{otherwise.} \end{cases} \tag{4.10}$$

Then ϕ is an update function for the Markov chain $(X_t)_{t\in\mathbb{N}}$.

4.3.4 A special case: from coins to dice

Assume that we are given a p-coin, and write $p_0 = 1-p$ and $p_1 = p$, to consider a fine and connected ladder of the form $\pi : (0,1) \to \Delta^k$ as in equation (4.4). In this

case the definition of neighbourhoods simplifies and so does the Markov chain defined in Proposition 4.15. Moreover, given the simplified structure of the state space, the update function defined in Proposition 4.18 is monotonic, so that monotonic CFTP can be employed. These observations are summarised in the following Corollary. Figure 4 gives a graphical representation of the dynamics of the Markov chain.

Corollary 4.19. *Let $\pi : (0,1) \to \Delta^k$ be a fine and connected ladder as in equation (4.4). The transition matrix of the Markov chain $(X_t)_{t\in\mathbb{N}}$ defined in Proposition 4.15 can be rewritten as*

$$P_{i,j} = \begin{cases} R_{i-1}\frac{1-p}{R_{i-1}\vee R_i} & \text{if } j = i-1, j > 0, \\ 1 - R_{i-1}\frac{1-p}{R_{i-1}\vee R_i} - R_{i+1}\frac{p}{R_i\vee R_{i+1}} & \text{if } j = i, \\ R_{i+1}\frac{p}{R_i\vee R_{i+1}} & \text{if } j = i+1, j < k, \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

Let $U \sim$ Unif$(0,1)$ and $p-$coin B be an independent r.v. (i.e. $\mathbb{P}(B=1) = 1-\mathbb{P}(B=0) = p$). The update function defined in Proposition 4.18 can be rewritten as

$$\phi(i,B,U) = \begin{cases} i-1 & \text{if } i > 0, B = 0, U \le \frac{R_{i-1}}{R_{i-1}\vee R_i}, \\ i+1 & \text{if } i < k, B = 1, U \le \frac{R_{i+1}}{R_i\vee R_{i+1}}, \\ i & \text{otherwise.} \end{cases} \tag{4.12}$$

Moreover, ϕ is a monotonic update function for the Markov chain $(X_t)_{t\in\mathbb{N}}$ where 0 and k are the minimum and maximum states respectively.

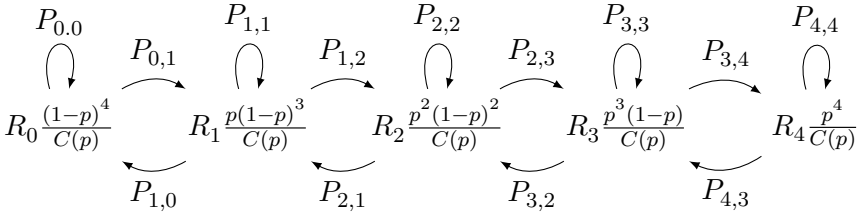


Figure 4. Transition probabilities on a fine and connected univariate ladder.

4.3.5 Efficiency of the algorithm

We now provide some results on the expected number of rolls of the original die required by CFTP and give insights on how the algorithm can be sped up. In

particular, we provide conditions for the expected number of rolls to be bounded uniformly in $\boldsymbol{p}$. Interestingly, this is always the case when $m = 1$, thus also in the Bernoulli Factory scenario. Moreover, we give tighter bounds when the univariate ladder is strictly log-concave, as defined below, and show that univariate ladders can be always transformed into an equivalent log-concave ladder through augmentation.

Definition 4.20 (Log-concave discrete distribution)**.** A discrete distribution $\boldsymbol{\mu}$ on $\Omega = \{0, \ldots, k\}$ is log-concave if for all $0 < i < k$,

$$\mu_i^2 \geq \mu_{i-1}\mu_{i+1}. \tag{4.13}$$

If the inequality is strict, then $\boldsymbol{\mu}$ is said to be strictly log-concave.

We shall also show that augmenting the degree of the ladder produces a log-concave distribution out of a ladder that is not itself log-concave. To that end, we provide the following general theorem that is of independent interest.

Theorem 4.21. *For every discrete random variable W on $\Omega = \{0, \ldots, n_0\}$ where $n_0 < \infty$, there exists a number $n = n(W)$ such that $W + B_n$ is strictly log-concave, where B_n is an independent Binomial$(n, 1/2)$.*

Proof of the theorem is deferred to the appendix. Several works have looked into whether log-concavity is preserved [21, 54], but checking whether some operations *introduce* log-concavity seems to be a harder problem [29] and the above result appears to be the first in this direction.

Building on Theorem 4.21, we will show that augmenting the degree of the ladder may lead to faster implementation. This is expanded in Proposition 4.26 and empirically verified in Examples 4.27 and 4.29.

General case

Proposition 4.22. *Let $\pi(\boldsymbol{p}) : \Delta^m \to \Delta^k$ be a fine and connected ladder of degree d. Write $\boldsymbol{n}_i$ as in Definition 4.5 and assume $E := \{b \in \{0, \ldots, m\} : \exists i, n_{i,b} = d\}$ is nonempty. Denote by N the number of rolls of the original die required by CFTP when the update function of Proposition 4.18 is used. Then, one can explicitly construct a new ladder $\pi'(\boldsymbol{p}) : \Delta^m \to \Delta^w$ of degree $2d$ where $w < \min\{k(m+1)^d, \binom{2d+m}{m}\}$ that is a disaggregation of π and such that*

$$\mathbb{E}[N] \leq \min_{b \in E} \frac{(ap_b)^{-2d} - 1}{1 - ap_b},$$

where $a \in (0,1]$ is a constant independent of $\boldsymbol{p}$.

If $E = \{0, \ldots, m\}$, then we can bound N by a quantity independent of $\boldsymbol{p}$, that is

$$\mathbb{E}[N] \leq \min_{b \in E} \frac{(ap_b)^{-2d} - 1}{1 - ap_b} \leq \frac{\left(\frac{a}{m+1}\right)^{-2d} - 1}{1 - \frac{a}{m+1}} \tag{4.14}$$

From coins to dice

We now restrict our analysis to rational functions of the form $f : (0,1) \to \Delta^v$, where an implementation of monotonic CFTP is possible. We study the efficiency of the proposed method in terms of the required number of tosses of the given p-coin. A direct consequence of Proposition 4.22 is the following.

Corollary 4.23. *Let $\pi(p) : (0,1) \to \Delta^k$ be a fine and connected ladder. Denote by N the number of tosses of the p-coin required by CFTP when the update function of Corollary 4.19 is used. Then, one can explicitly construct a new ladder $\pi'(\boldsymbol{p}) : (0,1) \to \Delta^{2k}$ that is a disaggregation of π and such that*

$$\mathbb{E}[N] \leq \min\left\{ \frac{(ap)^{-2(k-1)} - 1}{1 - ap}, \frac{(a(1-p))^{-2(k-1)} - 1}{1 - a(1-p)} \right\} \leq \frac{\left(\frac{a}{2}\right)^{-2(k-1)} - 1}{1 - \frac{a}{2}}$$

where $a = \min_i \left\{ \frac{R_{i-1}}{R_{i-1} \vee R_i} \wedge \frac{R_i}{R_{i-1} \vee R_i} \right\} \in (0,1]$ is a constant independent of $\boldsymbol{p}$.

Therefore, the expected running time of the algorithm can be bounded uniformly in p. However, the bound proposed in Corollary 4.23 is generally very loose and does not give insights into how the algorithm could potentially be sped up. We now provide a tighter bound under the condition that the ladder $\pi(p)$ is a log-concave distribution.

The proof of the following proposition is in spirit similar to the Path Coupling technique of [5].

Proposition 4.24. *Let $\pi : (0,1) \to \Delta^k$ be a univariate fine and connected ladder as in (4.4). Assume further that π is strictly log-concave and that the Markov chain and update function defined in Corollary 4.19 are used. Then*

$$\begin{aligned} \mathbb{P}(N \geq n) &\leq (k-1)\rho^n \qquad \text{and} \\ \mathbb{E}[N] &\leq \frac{(k-1)\rho}{1-\rho}, \end{aligned}$$

where $\rho \in (0,1)$ *for all* $p \in (0,1)$ *is given by*

$$\rho = \max_{i \in \{0,\dots,k-2\}} [1 - (P_{i,i+1} - P_{i+1,i+2}) - (P_{i+1,i} - P_{i,i-1})] \tag{4.15}$$

with $P_{i,j}$ *given by* (4.11) *with the convention that* $P_{k,k+1} = P_{0,-1} = 0$.

Remark 4.25. Given a univariate fine and connected ladder, ρ as in equation (4.15) can be explicitly computed for a fixed $p \in (0,1)$. Moreover, the algorithm still requires a finite number of tosses even if $p = 1$ or $p = 0$. In this case the expected number of tosses $\mathbb{E}[N]$ can be exactly computed:

$$\mathbb{E}[N] = \begin{cases} \frac{R_0 \vee R_1}{R_1} + \ldots + \frac{R_{k-2} \vee R_{k-1}}{R_{k-1}} & \text{if } p = 1, \\ \frac{R_0 \vee R_1}{R_0} + \ldots + \frac{R_{k-2} \vee R_{k-1}}{R_{k-2}} & \text{if } p = 0. \end{cases}$$

Given a rational function $f : (0,1) \to \Delta^v$, we have proved in Theorem 4.12 that it is always possible to construct a univariate fine and connected ladder $\pi : (0,1) \to \Delta^k$. However, π may not be strictly log-concave. Using Theorem 4.21, we now show that augmenting the ladder enough times produces a new ladder that is strictly log-concave, so that Proposition 4.24 applies.

Lemma 4.26. *Let* $\pi : (0,1) \to \Delta^k$ *be a univariate fine and connected ladder as in Section 4.2.2. Then, one can explicitly construct a new univariate fine and connected ladder* $\pi' : (0,1) \to \Delta^w$ *where* $w \geq k$ *such that* π' *is a disaggregation of* π *and* π' *is strictly log-concave.*

Hence increasing the degree of a generic univariate fine and connected ladder $\pi : (0,1) \to \Delta^k$ may lead to a faster implementation of monotonic CFTP despite an increased number of states that the chain needs to visit. Clearly, this leads to a trade-off that the user may want to calibrate, as shown in Examples 4.27 and 4.29.

Example 4.27. Consider sampling from the following ladder

$$\pi(p) \propto ((1-p)^4, 1000p(1-p)^3, p^2(1-p)^2, 500p^3(1-p), p^4).$$

Clearly, π is not log-concave. We can augment the ladder up to two times to obtain respectively

$$\begin{aligned} \pi^{(1)} &\propto ((1-p)^5, 1001p(1-p)^4, 1001p^2(1-p)^3, 501p^3(1-p)^2, 500p^4(1-p), p^5) \\ \pi^{(2)} &\propto ((1-p)^6, 1002p(1-p)^5, 2002p^2(1-p)^4, \\ &\quad 1502p^3(1-p)^3, 1001p^4(1-p)^2, 500p^5(1-p), p^6). \end{aligned}$$

Notice that now $\pi^{(2)}$ is strictly log-concave. Table 5 shows the empirical number of tosses required by the algorithm when sampling from either $\pi(p)$, $\pi^{(1)}(p)$ or $\pi^{(2)}(p)$ for different values of p. Notice that even if $\pi^{(1)}(p)$ is not log-concave, it still leads to a slightly faster implementation than when targeting $\pi^{(2)}(p)$.

	True value of p						
	0.01	0.1	0.25	0.5	0.75	0.9	0.99
$\widehat{\mathbb{E}}[N_{\pi}]$	561.31	621.73	827.86	1332.63	1433.59	1209.28	1090.54
$\widehat{\mathbb{E}}[N_{\pi^{(1)}}]$	92.35	12.21	7.72	8.65	9.48	12.59	87.05
$\widehat{\mathbb{E}}[N_{\pi^{(2)}}]$	93.17	13.33	9.43	11.29	11.67	14.47	89.56

Table 5: Average number of required tosses of the p-coin over 1,000 runs of the algorithm when targeting $\pi, \pi^{(1)}$ and $\pi^{(2)}$.

4.4 Examples and implementation

An `R` package implementing the method and reproducing the examples is available at `https://github.com/giuliomorina/DiceEnterprise`. The user is just required to define the function $f(\boldsymbol{p})$ and to provide a function that rolls the original die. Then, the package automatically constructs the fine and connected ladder and implements CFTP. If the original die has only two faces, the monotonic version of CFTP is automatically employed.

We now show how the method works and performs on some examples, all of which can be reproduced using the provided package. We start with a toy example to better explain and highlight the construction proposed in Theorem 4.12 and Proposition 4.15. Next, we examine efficiency of the monotonic and general versions of the algorithm by considering higher order rational functions. We also consider the so-called logistic Bernoulli factory as studied in [24]. We show that our method leads to a simple algorithm which on average requires the same number of tosses as the approach of [24]. Finally, we deal with a slightly different scenario where instead of an m-sided die, m independent coins are provided where the probability $\boldsymbol{p} = (p_0, \dots, p_{m-1}) \in (0,1)^m$ of tossing heads is unknown. In particular, we notice how we can construct a Dice Enterprise for the "Bernoulli Race" function considered by Dughmi et al. [12] which again has the same performance in terms of the expected number of required tosses.

Example 4.28 (Toy example of Bernoulli Factory)**.** Let $p \in (0,1)$ and assume we wish to generate a coin that lands heads with probability

$$\frac{\sqrt{2}p^3}{(\sqrt{2}-5)p^3+11p^2-9p+3},$$

having access only to a p-coin. Our proposed construction produces the following fine and connected ladder

$$\pi(p) = (3(1-p)^4, 3p(1-p)^3, 2p^2(1-p)^2, (\sqrt{2}+2)p^3(1-p), \sqrt{2}p^4),$$

via the following steps:

1. Let $C(p) = (\sqrt{2}-5)p^3+11p^2-9p+3$ and consider

$$f(p) = \frac{1}{C(p)}(\underbrace{-5p^3+11p^2-9p+3}_{D_0(p)}, \underbrace{\sqrt{2}p^3}_{D_1(p)}).$$

Convert $D_0(p)$ and $D_1(p)$ into homogeneous polynomials in the variables p and $(1-p)$ with positive coefficients and of the same degree. This can be achieved by using the multinomial theorem (cf. proof of Theorem 4.12). We get

$$\begin{aligned} D_0(p) &= 2p^2(1-p) + 3(1-p)^3, \\ D_1(p) &= \sqrt{2}p^3, \end{aligned}$$

so that we can equivalently consider the ladder

$$\pi'(p) = \frac{1}{C(p)}(3(1-p)^3, 2p^2(1-p), \sqrt{2}p^3).$$

and notice that if $X \sim \pi'$, then $W = \mathbb{I}(X \in \{3\})$ is distributed as $f(p)$.

2. Notice that π' is not a connected ladder, as there is no term proportional to $p(1-p)^2$. By applying the binomial theorem, we can construct a new ladder

$$\widetilde{\pi}(p) = \frac{1}{C(p)}(\widetilde{\pi}_0(p), \widetilde{\pi}_1(p), \widetilde{\pi}_2(p), \widetilde{\pi}_3(p), \widetilde{\pi}_4(p), \widetilde{\pi}_5(p)),$$

where

$$\begin{aligned}
\widetilde{\pi}_0(p) &= \pi'_0(p)\binom{1}{0}(1-p) = 3(1-p)^4,\\
\widetilde{\pi}_1(p) &= \pi'_0(p)\binom{1}{1}p = 3p(1-p)^3,\\
\widetilde{\pi}_2(p) &= \pi'_1(p)\binom{1}{0}(1-p) = 2p^2(1-p)^2,\\
\widetilde{\pi}_3(p) &= \pi'_1(p)\binom{1}{1}p = 2p^3(1-p),\\
\widetilde{\pi}_4(p) &= \pi'_2(p)\binom{1}{0}(1-p) = \sqrt{2}p^3(1-p),\\
\widetilde{\pi}_5(p) &= \pi'_2(p)\binom{1}{1}p = \sqrt{2}p^4.
\end{aligned}$$

Notice that if $Y \sim \widetilde{\pi}$, then $X = \mathbb{I}(Y \in \{2,3\}) + 2 \cdot \mathbb{I}(Y \in \{4,5\})$ is distributed as π'.

3. Finally, we can construct a fine and connected ladder by adding up together the terms where the same monomial appears:

$$\pi(p) = \frac{1}{C(p)}(3(1-p)^4, 3p(1-p)^3, 2p^2(1-p)^2, (\sqrt{2}+2)p^3(1-p), \sqrt{2}p^4).$$

Assume $Z \sim \pi$, $U \sim \text{Unif}(0,1)$ and let

$$\begin{aligned}
Y = {} & \mathbb{I}(Z=1) + 2 \cdot \mathbb{I}(Z=2) + \\
& 3 \cdot \mathbb{I}\left(Z=3, U \leq \frac{2}{2+\sqrt{2}}\right) + 4 \cdot \mathbb{I}\left(Z=3, U > \frac{2}{2+\sqrt{2}}\right) + 5 \cdot \mathbb{I}(Z=4)
\end{aligned}$$

so that $Y \sim \widetilde{\pi}$.

Table 6 shows the performance of CFTP for different values of the unknown probability p.

p	0.01	0.25	0.5	0.75	0.99
$f(p)$	0.00	0.02	0.22	0.65	0.99
$\widehat{f}(p)$	$_{0.00}0.00_{\,0.00}$	$_{0.01}0.01_{\,0.02}$	$_{0.18}0.21_{\,0.23}$	$_{0.63}0.66_{\,0.69}$	$_{0.98}0.99_{\,0.99}$
$\widehat{\mathbb{E}}[N]$	4.80	7.45	10.61	8.05	5.94

Table 6: Implementation of the Bernoulli Factory for the function $f(p) = \frac{\sqrt{2}p^3}{(\sqrt{2}-5)p^3+11p^2-9p+3}$ and for different values of the true unknown probability p. The algorithm has been run 1,000 times to obtain tosses of the $f(p)$-coin and $\widehat{f}(p)$ is the sample average. Smaller numbers represent 95% confidence interval computed via the method of [57]. $\widehat{\mathbb{E}}[N]$ is the empirical expected number of tosses in a run of the CFTP algorithm.

Example 4.29 (Augmenting the number of states can lead to faster running time)**.** Given a p-coin, consider constructing a 3-sided die where the probability of rolling each face is given by

$$\pi(p) \propto \{p^{20}, p^{10}(1-p)^{10}, (1-p)^{20}\}. \tag{4.16}$$

A naive rejection sampling approach to construct a Bernoulli Factory for $\pi(p)$ would be the following: toss the p-coin 20 times and with probability 1/3 output 1 if all the tosses are heads, with probability 1/3 output 2 if the first 10 tosses are heads and the last 10 tosses are tails, with probability 1/3 output 3 if all the tosses are tails. In all other cases, restart the algorithm.

Assume now that $p = 1/2$, so that the expected number of tosses of this naive procedure would be $\mathbb{E}[N] = 2^{20} \approx 10^6$. Table 7 shows the performance of our novel algorithm on the same example when targeting the ladder of equation (4.16), as well as when targeting the augmented ladder where extra states are added. Indeed, Lemma 4.26 and Proposition 4.24 suggest that doing so may lead to faster performance, as empirically confirmed. Notice that to get a strictly log-concave ladder, we need to augment π at least 203 times. In practice, it is enough to augment it around 40 times to obtain optimal performance, due to the trade-off effect discussed in Section 4.3.5.

	Number of states added to the original ladder $\pi(p)$						
	+0	+20	+40	+60	+80	+100	+120
$\widehat{\mathbb{E}}[N]$	5337.7	585.7	471.4	481.7	529.3	590.4	647.9
	+140	+160	+180	+200	+220	+240	+260
$\widehat{\mathbb{E}}[N]$	717.2	774.2	840.2	892.3	927.3	996.4	1038.9

Table 7: Implementation of the Dice Enterprise for the function of eq. (4.16) when $p = 1/2$. The algorithm has been run 1,000 times and $\widehat{\mathbb{E}}[N]$ is the empirical number of tosses of the p-coin required. The augmented ladder is strictly log-concave when at least 203 states are added.

Consider now a slightly different example, where a 3-sided fair die is given, i.e. $\boldsymbol{p} = (1/3, 1/3, 1/3)$, and the aim is to construct a 4-sided die where the probability of rolling each face is given by

$$\pi(\boldsymbol{p}) \propto \{p_0^5 p_1^5 p_2^5, p_0^{15}, p_1^{15}, p_2^{15}\}. \tag{4.17}$$

A naive approach as the one before would require on average $\mathbb{E}[N] = 3^{15} \approx 1.4 \times 10^7$ rolls of the $\boldsymbol{p}$-die. Although the result of Proposition 4.24 does not hold here, as a monotonic implementation of CFTP is not possible, augmenting the ladder may still lead to faster performance. This is indeed the case, as shown in Table 8, where targeting the ladder with 60 extra states leads to an implementation that requires on average around 840 tosses of the $\boldsymbol{p}$-die, instead of more than 100,000 when directly targeting the original $\pi(\boldsymbol{p})$.

	Number of states added to the original ladder $\pi(\boldsymbol{p})$							
	+0	+10	+20	+30	+40	+50	+60	+70
$\widehat{\mathbb{E}}[N]$	174246.4	2569.0	1341.5	1032.9	912.5	874.0	841.4	860.1

Table 8: Implementation of the Dice Enterprise for the function of eq. (4.17) when $\boldsymbol{p} = (1/3, 1/3, 1/3)$. The algorithm has been run 1,000 times and $\widehat{\mathbb{E}}[N]$ is the empirical number of rolls of the $\boldsymbol{p}$-die required.

Example 4.30 (Logistic Bernoulli Factory)**.** Consider constructing a Bernoulli factory for the function

$$\frac{Cp}{1 + Cp}, \qquad C > 0.$$

Such problem is considered in [24] where it is referred as constructing a logistic

Bernoulli factory. In the same paper, the author proposes an ad-hoc algorithm that exploits properties of thinned Poisson processes and requires on average $\mathbb{E}[N_H] = C/(1+Cp)$ tosses of the p-coin. We now show that our proposed method leads to an alternative algorithm that requires on average the same number of tosses. The fine and connected ladder for this target is

$$\pi(p) = \frac{1}{1+Cp}((1+C)p, (1-p)).$$

Given $Y \sim \pi$ and $U \sim \text{Unif}(0,1)$, we output heads if $Y = 1, U < C/(1+C)$ and tails otherwise. Sampling from $\pi(p)$ boils down to sampling from the stationary distribution of a Markov chain consisting of only two states, as depicted in Figure 5. CFTP needs to keep track of only two chains starting in the two states and the algorithm stops as soon as one of the two chain moves, as they cannot both move at the same time. In particular, the particles coalesce if heads is tossed or if the uniform r.v. U drawn by the algorithm is such that $U \leq 1/(1+C)$. Therefore, CFTP is equivalent to algorithm 18 which is a special case of the 2-coin algorithm presented in [16, 17] with $c_1 = C, c_2 = 1, p_1 = p, p_2 = 1$.

Algorithm 18 Logistic Bernoulli Factory

Input: black box to sample from $\text{Ber}(p)$, a constant $C > 0$.

Output: a sample from $\text{Ber}(Cp/(1+Cp))$.

1: Sample $U \sim \text{Unif}(0,1)$
2: **if** $U \leq \frac{1}{1+C}$ **then** set $Y := 0$
3: **else**
4: Sample $B \sim \text{Bern}(p)$
5: **if** $B = 1$ **then** set $Y := 1$
6: **else** discard U, B and GOTO 1
7: **end if**
8: **end if**
9: Output Y

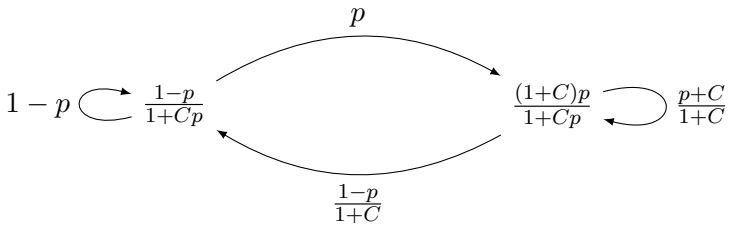


FIGURE 5. Dynamic of the Markov chain with stationary distribution $\pi(p) = \frac{1}{1+Cp}((1+C)p, (1-p))$.

The probability that the algorithm stops at a specific iteration is $\frac{1+Cp}{1+C}$. Since each iteration is independent of the others and the probability that a toss of the p-coin is required is $\frac{C}{1+C}$, the average number of tosses is then given by

$$\mathbb{E}[N_{\text{CFTP}}] = \frac{1+C}{1+Cp}\frac{C}{1+C} = \frac{C}{1+Cp}$$

and it is thus equal to $\mathbb{E}[N_H]$.

Example 4.31 (Independent coins and Bernoulli Race)**.** We now deal with a slightly different scenario where instead of having access to a die, m independent coins are given. Similarly, the probability of tossing heads on each of the coin is unknown and given by $\boldsymbol{p} = (p_0, \dots, p_{m-1}) \in (0,1)^m$, so that the problem is now obtaining a sample from a rational function $f : (0,1)^m \to \Delta^v$. There are several ways to transform tosses of m coins into a roll of a die. In particular, we can construct an $(m+1)$-sided die in the following fashion. Firstly, we choose uniformly which coin to toss, say the i^{th}. If the result is heads we output $i \in \{0, \dots, m-1\}$, otherwise we output m. The probabilities of obtaining each face by rolling the so constructed $(m+1)$-sided die is then given by $\widetilde{\boldsymbol{p}} = \left(\frac{p_0}{m}, \dots, \frac{p_{m-1}}{m}, 1 - \frac{1}{m}\sum_{i=0}^{m-1} p_i\right) \in \Delta^m$. The function $f(\boldsymbol{p})$ can be transformed into a function of $\widetilde{\boldsymbol{p}}$ by substituting $p_i = m\widetilde{p}_i$.

We now consider the function $f(\boldsymbol{p}) = \frac{1}{\sum_{i=1}^m p_i}(p_1, \dots, p_m)$ as in [12], where the problem of tossing such $f(\boldsymbol{p})$-die is named Bernoulli Race. Their proposed algorithm requires on average $\mathbb{E}[N_D] = m/\sum_{i=1}^m p_i$ tosses of the m coins. After applying the required variable transformation, we then consider $f(\widetilde{\boldsymbol{p}}) = \frac{1}{\sum_{i=0}^{m-1} \widetilde{p}_i}(\widetilde{p}_0, \dots, \widetilde{p}_m)$ and we can then employ our Dice Enterprise methodology. In this particular problem, $f(\widetilde{\boldsymbol{p}})$ is already a multivariate ladder and the transition matrix of the chain constructed as in Proposition 4.15 is given by

$$P = \begin{pmatrix} 1 - \sum_{i \neq 0} \widetilde{p}_i & \widetilde{p}_1 & \widetilde{p}_2 & \dots & \widetilde{p}_m \\ \widetilde{p}_0 & 1 - \sum_{i \neq 1} \widetilde{p}_i & \widetilde{p}_2 & \dots & \widetilde{p}_m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \widetilde{p}_0 & \widetilde{p}_1 & \widetilde{p}_2 & \dots & 1 - \sum_{i \neq m} \widetilde{p}_i \end{pmatrix}$$

Notice that CFTP terminates as soon as either $0, 1, \dots, m-1$ is rolled and continues only when the outcome of the roll is the m^{th} face. In this case, each iteration of CFTP is independent of the other and the probability that the algorithm stops is given by

$$\mathbb{P}(\text{CFTP stops}) = 1 - \frac{1}{m} \sum_{i=1}^{m} \mathbb{P}(\text{i}^{\text{th}} \text{ coin returns tails}) = \frac{1}{m} \sum_{i=0}^{m-1} p_i$$

so that $\mathbb{E}[N_{CFTP}] = m / \sum_{i=0}^{m-1} p_i$ and the algorithm is actually equivalent to the one proposed in [12].

4.5 Discussion

The Dice Enterprise algorithm introduced in this chapter is a generalisation of the celebrated Bernoulli Factory algorithms to rational mappings of categorical distributions. It offers a fully automated procedure that does not require further user intervention or case specific design tweaks. Furthermore, in the "coin to dice" case the efficiency of the algorithm can be automatically boosted by increasing the degree of the target polynomials until the distribution is log-concave which guarantees fast convergence. The version we developed is based on Coupling From the Past and enjoys an efficient monotonic implementation in the "coin to dice" case, however CFTP can be replaced by any other Markov chain perfect sampling routine, including Fill's interruptible algorithm. We demonstrated that several specialised Bernoulli factory algorithms introduced in literature, such as the two coin algorithm, the logistic Bernoulli factory or the Bernoulli race can be regarded as special versions of the Dice Enterprise. A natural open problem that follows from this chapter is to design a monotone version of the Dice Enterprise in the "dice to dice" scenario. Further studies may also look into providing bounds for the degree of the decomposition of rational functions into ladders (based on Pólya positive homogeneous polynomial theorem [50, 51]) or the number of Bernoulli trials needed to introduce log-concavity when convoluted with a discrete random variable. Understanding of these questions is necessary for obtaining more precise upper bounds on running

time of the algorithm. Computing lower bounds for the expected number of rolls required by CFTP – perhaps through information criteria (c.f. [38]) or building on [7, 26] - would complement the theoretical analysis.

Another question of particular interest is establishing the relation between our approach and that of [42]. Both theirs and our work builds on Polya's theorem on homogeneous polynomials, which ensures positivity of all the coefficients and therefore allows for a construction of an equivalent probability distribution in a form that is amenable to simulation via a carefully designed finite automaton, or Markov chain, respectively. The focus of [42] is on the theoretical side of characterising distributions through automata rather than on practical algorithm design or efficiency. In particular, the block simulation considered in their Theorem 2.2, and outlined in Proposition 2.5, is closely related to what we termed the naive rejection sampling approach in Example 4.29. Its cost would scale exponentially in the degree of the involved polynomials. On the other hand, [42] poses an open question (Problem 4.1) and asks what is the smallest size of an automaton that simulates $f(p)$, and how to find it. While we do not know the answer to this problem, we conjecture that when f is a rational function with coefficients in $\mathbb{Q}$, the CFTP procedure we designed for simulating a fine and connected ladder $\pi(\boldsymbol{p}) : \Delta^m \to \Delta^k$, is a finite automaton with the set of states $S = \{0, \ldots, k\} \times \{0, \ldots, k\}$ and alphabet $\{0, \ldots, m\}$. We believe investigating systematically the connections between these approaches is a research direction that may lead to interesting conclusions.

4.6 Proofs of the results of Chapter 4

Proof of Proposition 4.11

By construction π' is a fine ladder on $\Omega' = \{0, \dots, w\}$. Since one augmentation operation yields a ladder sampling from which is equivalent to sampling from π, so does d-fold augmentation. It remains to show that π' is connected. To this end note that each state of π' is of the form $C\pi_i(\boldsymbol{p})\binom{d}{\boldsymbol{n}}\prod_{l=0}^{m} p_l^{n_l}$ for some constant C, some $i \in \{0, ..., k\}$ and some $\boldsymbol{n} \in \Lambda_d^m$. Define sets $A_0, \dots, A_k$ as

$$A_i = \{a \in \Omega' : \pi'_a(\boldsymbol{p}) = C_a \pi_i(\boldsymbol{p}) \binom{d}{\boldsymbol{n}} \prod_{l=0}^{m} p_l^{n_l} \text{ for some } \boldsymbol{n} \in \Lambda_d^m,\ C_a \in \mathbb{R}\}. \quad (4.18)$$

First notice that for a fixed i all the states in A_i are connected by construction due to $d-$fold augmentation. It is then enough to show that $A_i \cap A_j \neq \emptyset$ for all $j \neq i$. Indeed, let $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ be the degree of $\pi_i(\boldsymbol{p})$ and $\pi_j(\boldsymbol{p})$ respectively. Then, the numerators of $\pi_j(\boldsymbol{p})\binom{d}{\boldsymbol{n}_i}\prod_{l=0}^{m} p_l^{n_{i,l}}$ and $\pi_i(\boldsymbol{p})\binom{d}{\boldsymbol{n}_j}\prod_{l=0}^{m} p_l^{n_{j,l}}$ have the same degree $\boldsymbol{n}_i + \boldsymbol{n}_j$ and the respective state $a \in \Omega'$ with probability $\pi_{a'}(\boldsymbol{p})$ of degree $\boldsymbol{n}_i + \boldsymbol{n}_j$ satisfies $a \in A_i \cap A_j$. □

Lemma 4.32 (Pólya [50])**.** *Let $f : \Delta^m \to \mathbb{R}$ be a homogeneous and positive polynomial in the variables $p_0, \dots, p_m$, i.e. all the monomials of the polynomial have the same degree. Then, for all sufficiently large n, all the coefficients of $(p_0 + \dots + p_m)^n f(p_0, \dots, p_m)$ are positive.*

Lemma 4.33. *Let $f : \Delta^m \to (0, 1)$ be a rational function over $\mathbb{R}$. Then, there exist homogeneous polynomials*

$$d(\boldsymbol{p}) = d(p_0, \dots, p_m) = \sum_{\boldsymbol{n} \in \Lambda_d^m} d_{\boldsymbol{n}} \prod_{j=0}^{m} p_j^{n_j},$$

$$e(\boldsymbol{p}) = e(p_0, \dots, p_m) = \sum_{\boldsymbol{n} \in \Lambda_d^m} e_{\boldsymbol{n}} \prod_{j=0}^{m} p_j^{n_j},$$

where $d_{\boldsymbol{n}}$ and $e_{\boldsymbol{n}}$ are real coefficients such that $0 \leq d_{\boldsymbol{n}} \leq e_{\boldsymbol{n}}$ and $f(\boldsymbol{p}) = d(\boldsymbol{p})/e(\boldsymbol{p})$. We will refer to d as the degree of the decomposition.

Proof. The lemma is a variation of Lemma 2.7 of [42], where $m = 1$ and coefficients are integers, and the proof follows the reasoning therein.

As $f(\boldsymbol{p})$ is a rational function, it may be written as

$$f(\boldsymbol{p}) = \frac{\overline{D}(\boldsymbol{p})}{\overline{E}(\boldsymbol{p})},$$

and we can assume that $\overline{D}(\boldsymbol{p})$ and $\overline{E}(\boldsymbol{p})$ are relatively prime polynomials. Since $f(\boldsymbol{p}) \in (0,1)$ for all $\boldsymbol{p} \in \Delta^m$ and $\overline{D}(\boldsymbol{p})$ does not share any common root with $\overline{E}(\boldsymbol{p})$, it follows that $\overline{D}(\boldsymbol{p})$ and $\overline{E}(\boldsymbol{p})$ do not change sign in Δ^m so that we can assume without loss of generality that $\overline{D}(\boldsymbol{p})$ and $\overline{E}(\boldsymbol{p})$ are positive polynomials. Let d_0 be the maximum degree of the polynomials $\overline{D}(\boldsymbol{p})$ and $\overline{E}(\boldsymbol{p})$. A general representation of the polynomials is given by

$$\overline{D}(\boldsymbol{p}) = \sum_{i=0}^{d} \sum_{\boldsymbol{n} \in \Lambda_i^m} a_{\boldsymbol{n}} \prod_{j=0}^{m} p_j^{n_j}, \qquad \overline{E}(\boldsymbol{p}) = \sum_{i=0}^{d} \sum_{\boldsymbol{n} \in \Lambda_i^m} b_{\boldsymbol{n}} \prod_{j=0}^{m} p_j^{n_j}.$$

Notice that in general $\overline{D}(\boldsymbol{p})$ and $\overline{E}(\boldsymbol{p})$ are not homogeneous polynomials, but it is possible to increase the degree of each term of the summation to be equal to d_0. In fact, since $p_0 + \ldots + p_m = 1$, one can use the multinomial theorem to define homogeneous polynomials $D(\boldsymbol{p})$ and $E(\boldsymbol{p})$ as

$$\begin{aligned}
\overline{D}(\boldsymbol{p}) &= \sum_{i=0}^{d} \sum_{\boldsymbol{n} \in \Lambda_i^m} a_{\boldsymbol{n}} (p_0 + \ldots + p_m)^{d-i} \prod_{j=0}^{m} p_j^{n_j} \\
&= \sum_{i=0}^{d} \sum_{\boldsymbol{n} \in \Lambda_i^m} \sum_{\boldsymbol{n}' \in \Lambda_{d-i}^m} a_{\boldsymbol{n}} \binom{d-i}{\boldsymbol{n}'} \prod_{j=0}^{m} p_j^{n_j + n_j'} \\
&= \sum_{\boldsymbol{n} \in \Lambda_d^m} d_{\boldsymbol{n}} \prod_{j=0}^{m} p_j^{n_j} =: D(\boldsymbol{p}),
\end{aligned}$$

where

$$d_{\boldsymbol{n}} = \sum_{i=0}^{d} \sum_{\widetilde{\boldsymbol{n}} \in \Lambda_i^m} \sum_{\boldsymbol{n}' \in \Lambda_{d-i}^m : \widetilde{\boldsymbol{n}} + \boldsymbol{n}' = \boldsymbol{n}} a_{\widetilde{\boldsymbol{n}}} \binom{d-i}{\boldsymbol{n}'}.$$

Analogously

$$\overline{E}(\boldsymbol{p}) = \sum_{i=0}^{d} \sum_{\boldsymbol{n} \in \Lambda_i^m} b_{\boldsymbol{n}} (p_0 + \ldots + p_m)^{d-i} \prod_{j=0}^{m} p_j^{n_j} = \sum_{\boldsymbol{n} \in \Lambda_d^m} e_{\boldsymbol{n}} \prod_{j=0}^{m} p_j^{n_j} := E(\boldsymbol{p}).$$

Notice that $D(\boldsymbol{p})$ and $E(\boldsymbol{p})$ are positive polynomials. Moreover, since $f(\boldsymbol{p}) < 1$,

it follows that also $E(\boldsymbol{p}) - D(\boldsymbol{p})$ is a positive polynomial. Therefore, by Lemma 4.32 there exists a sufficiently large n, such that the polynomials $d(\boldsymbol{p}) = (p_0 + \ldots + p_m)^n D(\boldsymbol{p})$, $e(\boldsymbol{p}) = (p_0 + \ldots + p_m)^n E(\boldsymbol{p})$ and $e(\boldsymbol{p}) - d(\boldsymbol{p})$, all have positive coefficients. Hence, as required, $0 \leq d_{\boldsymbol{n}} \leq e_{\boldsymbol{n}}$ and

$$f(\boldsymbol{p}) = \frac{\overline{D}(\boldsymbol{p})}{\overline{E}(\boldsymbol{p})} = \frac{D(\boldsymbol{p})}{E(\boldsymbol{p})} = \frac{d(\boldsymbol{p})}{e(\boldsymbol{p})}. \tag{4.19}$$

The degree of the decomposition is therefore $d = d_0 + n$. □

Proof of Theorem 4.12

Since $f(\boldsymbol{p}) = (f_0(\boldsymbol{p}), \ldots, f_v(\boldsymbol{p}))$ is a rational function, we can apply Lemma 4.33 to each $f_i(\boldsymbol{p})$ and write

$$f(\boldsymbol{p}) = \left(\frac{d_0(\boldsymbol{p})}{e_0(\boldsymbol{p})}, \frac{d_1(\boldsymbol{p})}{e_1(\boldsymbol{p})}, \ldots, \frac{d_v(\boldsymbol{p})}{e_v(\boldsymbol{p})} \right).$$

Let $C(\boldsymbol{p})$ be the lowest common multiple of the denominators $e_i(\boldsymbol{p})$ and express $f(\boldsymbol{p})$ as

$$f(\boldsymbol{p}) = \frac{1}{C(\boldsymbol{p})}(g_0(\boldsymbol{p}), \ldots, g_v(\boldsymbol{p})).$$

Assume w.l.o.g. that each polynomial $g_i(\boldsymbol{p})$ has degree d (if this is not the case, let d_i be the degree of $g_i(\boldsymbol{p})$ and multiply it by $(p_0 + \ldots + p_m)^{d-d_i}$) and write

$$\frac{g_i(\boldsymbol{p})}{C(\boldsymbol{p})} = \frac{1}{C(\boldsymbol{p})} \sum_{\boldsymbol{n} \in \Lambda_d^m} a_{i,\boldsymbol{n}} \prod_{j=0}^{m} p_j^{n_j}. \tag{4.20}$$

Having applied Lemma 4.33 it follows $a_{i,\boldsymbol{n}} \geq 0$ for all $i \in \{0, \ldots, v\}$, $\boldsymbol{n} \in \Lambda_d^m$. Therefore, we can construct a distribution $\pi' : \Delta^m \to \Delta^w$ on $\Omega' = \{0, \ldots, w\}$, where $w < (v+1)\binom{d+m}{m}$ and where each state is one term of the summation in (4.20) for a fixed i and thus of the form $\frac{1}{C(\boldsymbol{p})} a_{i,\boldsymbol{n}} \prod_{j=0}^{m} p_j^{n_j}$.

By construction π' is a disaggregation of f. Indeed, consider v sets $A_0, \ldots, A_v$ defined as

$$A_i = \{a \in \Omega' : \pi'_a(\boldsymbol{p}) = \frac{1}{C(\boldsymbol{p})} a_{i,\boldsymbol{n}} \prod_{j=0}^{m} p_j^{n_j} \text{ for a } \boldsymbol{n} \in \Lambda_d^m\}.$$

It then follows

$$f_i(\boldsymbol{p}) = \frac{g_i(\boldsymbol{p})}{C(\boldsymbol{p})} = \sum_{h \in A_i} \pi'_h(\boldsymbol{p}) = \frac{1}{C(\boldsymbol{p})} \sum_{\boldsymbol{n} \in \Lambda_d^m} a_{i,\boldsymbol{n}} \prod_{j=0}^{m} p_j^{n_j}.$$

By discarding any null term in $\pi'(\boldsymbol{p})$, it follows that π' is a multivariate ladder. Finally, via Proposition 4.11 we construct a fine and connected multivariate ladder $\pi : \Delta^m \to \Delta^k$ where $k < \min\{(w+1)(m+1)^d, \binom{2d+m}{m}\}$, such that sampling from each f, π' and π is equivalent.

□

Proof of Proposition 4.15

We shall prove the result by showing that P is a stochastic matrix and that the detailed balance condition is satisfied for all $\boldsymbol{p} \in \Delta^m$. Recall that the off-diagonal elements of P are given by the off-diagonal elements of $V \circ W$ where $\circ$ denotes the entrywise product, W is defined in equation (4.8) and V is the output of Algorithm 17. We first prove that

$$\sum_{j \in \mathcal{N}_b(i)} V_{i,j} \leq 1, \quad \forall b \in \{0, \dots, m\}, i \in \Omega.$$

Notice that by how the weights $\mathcal{W}_b(i)$ are defined within the algorithm, we have $\sum_{j \in \mathcal{N}_b(i)} V_{i,j} = \mathcal{W}_b(i)$.

Having fixed i and b, assume that one of the $V_{i,j}$, where $j \in \mathcal{N}_b(i)$, is obtained in line 6 of the algorithm. Denote by $\mathcal{W}_b^\star(i)$ the new value of $\mathcal{W}_b(i)$ after it has been updated for all $j \in \mathcal{N}_b(i)$. It follows

$$\mathcal{W}_b^\star(i) = \mathcal{W}_b(i) + \sum_{j \in \mathcal{N}_b(i)} \frac{R_j}{\mathcal{S}_b(i)} = \mathcal{W}_b(i) + \sum_{j \in \mathcal{N}_b(i)} \frac{R_j}{\sum_{h \in \mathcal{N}_b(i)} R_h}(1 - \mathcal{W}_b(i)) = 1,$$

where the value of $S_b(i)$ is given in line 9 of the algorithm. At this point the algorithm has assigned a value to $V_{i,j}$ for all $j \in \mathcal{N}_b(i)$ and thus $\sum_{j \in \mathcal{N}_b(i)} V_{i,j} = \mathcal{W}_b^\star(i) = 1$.

Assume now that all the $V_{i,j}$ for $j \in \mathcal{N}_b(i)$ have been assigned in line 8 of the algorithm. For fixed i, we then have that $j \in \mathcal{N}_b(i)$ and let $d \in \{0, \dots, m\}$ such that $i \in \mathcal{N}_d(j)$. Then $V_{j,i}$ is assigned in line 6 of the algorithm. Denote the new value of $\mathcal{W}_b(i)$ assigned in line 8 of the algorithm as $\mathcal{W}_b^\star(i)$. It follows

$$\mathcal{W}_b^\star(i) = \mathcal{W}_b(i) + \frac{R_j}{\mathcal{S}_d(j)} \leq \mathcal{W}_b(i) + \frac{R_j}{\mathcal{S}_b(i)} \leq \mathcal{W}_b(i) + \sum_{j \in \mathcal{N}_b(i)} \frac{R_j}{\mathcal{S}_b(i)} = 1,$$

where the fact that $\mathcal{S}_d(j) \geq \mathcal{S}_b(i)$ follows from the fact that b and i are chosen in line 4 of the algorithm to maximise $\mathcal{S}_b(i)$. The value of $\mathcal{W}_b(i)$ will then always be less or equal than 1, so that $\sum_{j\in\mathcal{N}_b(i)} V_{i,j} = \mathcal{W}_b(i) \leq 1$.

We then have

$$\sum_{\substack{j=0\\ j\neq i}}^{k} P_{i,j} = \sum_{b=0}^{m} \sum_{j\in\mathcal{N}_b(i)} V_{i,j} p_b \leq \sum_{b=0}^{m} p_b = 1,$$

as required. It is now enough to prove that $\pi(\boldsymbol{p})$ satisfies the detailed balance condition for all $\boldsymbol{p} \in \Delta^m$. If $j \notin \mathcal{N}(i)$, then $P_{i,j} = P_{j,i} = 0$ and the balance condition is trivially satisfied. For $j \in \mathcal{N}(i)$ we have

$$\frac{\pi_j(\boldsymbol{p})}{\pi_i(\boldsymbol{p})} = \frac{R_j \prod_{h=0}^{m} p_h^{n_{j,h}}}{R_i \prod_{h=0}^{m} p_h^{n_{i,h}}} = \frac{R_j p_b}{R_i p_c},$$

and by equation (4.8), $W_{i,j}/W_{j,i} = p_b/p_c$. The fact that $V_{i,j}/V_{j,i} = R_j/R_i$ follows directly from how these values are assigned in the algorithm for the pair i, j in lines 6 and 8. Given the connectedness condition, $\pi(\boldsymbol{p})$ is also the unique limiting distribution. □

Proof of Proposition 4.16

By contradiction, assume that there exists a different reversible Markov chain with transition matrix Q that has the same adjacency structure and stationary distribution as the P-chain, and such that $Q \succeq_P P$. It follows that also Q has a similar decomposition as in equation (4.5) and the off-diagonal elements of Q will be the same as the entries of $\widetilde{V} \circ W$, where $\circ$ denotes the entrywise product and with W as in equation (4.8), while $\widetilde{V}$ is a matrix of real numbers. Since $Q \succeq_P P$ and $Q \neq P$, there must exist indices i, j such that $\widetilde{V}_{i,j} > V_{i,j}$. We distinguish two cases:

- The value of $V_{i,j}$ is assigned in line 6 of Algorithm 17. Then, let $b \in \{0, \ldots, m\}$ such that $j \in \mathcal{N}_b(i)$ and notice that by how the algorithm is designed we have $\sum_{j\in\mathcal{N}_b(i)} V_{i,j} = 1$ (cf. proof of Proposition 4.15). Therefore $\sum_{j\in\mathcal{N}_b(i)} \widetilde{V}_{i,j} > 1$. We reach a contradiction by observing

$$\sum_{\substack{j=0\\ j\neq i}}^{k-1} Q_{i,j} = \sum_{c=0}^{m} \sum_{j\in\mathcal{N}_c(i)} \widetilde{V}_{i,j} p_c \xrightarrow{p_b \to 1} \sum_{j\in\mathcal{N}_b(i)} \widetilde{V}_{i,j} > 1.$$

- The value of $V_{i,j}$ is assigned in line 8 of Algorithm 17. Since the Q-chain is reversible, it follows that also $\widetilde{V}_{j,i} > V_{j,i}$. However, the value of $V_{j,i}$ is assigned

in line 6 of the algorithm and we reach the same contradiction as before.

□

Proof of Proposition 4.18

Fix a state $i \in \Omega$ and notice that if $j \notin \mathcal{N}(i)$, then $\mathbb{P}(\phi(i, B, U) = j) = P_{i,j} = 0$. For any outcome $b \in \{0, \dots, m\}$ on the die, recall $\mathcal{N}_b(i) = \{j_0, \dots, j_w\}$, is the set of states accessible from i. It follows for any $j_l \in \mathcal{N}_b(i)$ that

$$\begin{aligned}\mathbb{P}(\phi(i, B, U) = j_l) &= \mathbb{P}\left(B = b, \sum_{h=0}^{l-1} V_{i,j_h} < U \leq \sum_{h=0}^{l} V_{i,j_h}\right) \\ &= p_b \mathbb{P}\left(U \leq V_{i,j_l}\right) = P_{i,j_l}.\end{aligned}$$

Hence, ϕ is an update function for the Markov chain $(X_t)_{t\in\mathbb{N}}$. □

Proof of Corollary 4.19

Given a fine and connected ladder $\pi : (0, 1) \to \Delta^k$ as in equation (4.4), for $1 \leq i \leq k - 1$, we have

$$\begin{aligned}&\mathcal{N}_0(i) = \{i - 1\}, \qquad \mathcal{N}_1(i) = \{i + 1\}, \qquad \mathcal{N}(i) = \{i - 1, i + 1\}, \\ &\mathcal{S}_0(i) = R_{i-1}, \qquad \mathcal{S}_1(i) = R_{i+1}.\end{aligned}$$

Then, the matrix W defined in equation (4.8) and the off-diagonal entries of the matrix V output by Algorithm 17 are given by

$$W_{i,j} = \begin{cases} p & \text{if } j = i + 1 \\ (1 - p) & \text{if } j = i - 1 \\ 0 & \text{otherwise} \end{cases}, \qquad V_{i,j} = \begin{cases} \frac{R_{i+1}}{R_i \vee R_{i+1}} & \text{if } j = i + 1 \\ \frac{R_{i-1}}{R_{i-1} \vee R_i} & \text{if } j = i - 1 \\ 0 & \text{otherwise} \end{cases}$$

Therefore, the transition matrix P defined in (4.5) is equivalent to (4.11) and the update function defined in (4.10) is the same as (4.12).

To see why ϕ is a monotonic update function, consider $i \leq j$. It is trivial to check that $\phi(i, B, U) \leq \phi(j, B, U)$ if $j \neq i + 1$. If $j = i + 1$, the monotonic condition would not be satisfied only if $\phi(i, B, U) = i+1$ and $\phi(i+1, B, U) = i$. However, this can not happen as it would require B to be equal to 0 and 1 simultaneously. □

Proof of Theorem 4.21

Proof. Let $w_0, \ldots, w_{n_0}$ be the probabilities of W on $\Omega = \{0, \ldots, n_0\}$. We shall consider generating function $P(x) = \sum_{i=0}^{n_0} w_i x^i$. This function is a product of linear and quadratic functions, that is

$$P(x) = c \prod_{j=1}^{k_0} ((x - a_j)^2 + b_j^2) \prod_{l=1}^{l_0} (x + c_l),$$

where $b_j \neq 0$ and $c_l > 0$ (the latter follows from the fact that a polynomial with positive coefficients cannot have positive roots). Now, it suffices to show that for big n the sequences of coefficients generated by

$$Q_n(x) = ((x - a)^2 + b^2)(1 + x)^n, \quad L_n(x) = (x + c)(1 + x)^n, \text{ where } b \neq 0, c > 0,$$

is positive and log concave. Indeed, since convolution preserves positivity and log-concavity, and corresponds to summing random variables, we can find suitable binomial $B(n, 1/2)$ (whose generating function is precisely $\frac{1}{2^n}(1 + x)^n$) for each factor of P separately. Note that we ignore normalizing constants, as positivity and log-concavity are not affected.

The rest is just an attempt to verify this. In case of L_n there is nothing to prove since the sequence generated by $x + c$ with $c > 0$ is $(c, 1, 0, \ldots)$ and it is positive and log-concave itself. Since $(x - a)^2 + b^2 = x^2 - 2ax + a^2 + b^2$, the sequence generated by Q_n is

$$a_k = (a^2 + b^2)\binom{n}{k} - 2a\binom{n}{k-1} + \binom{n}{k-2}, \qquad k \geq 0.$$

Here we adapt the notation $\binom{n}{k} = 0$ for $k < 0$ and $k > n$. We first show that for big n this sequence is non-negative. The inequality $a_k \geq 0$ is equivalent to

$$(a^2 + b^2)(n - k + 1)(n - k + 2) - 2ak(n - k + 2) + k(k - 1) \geq 0.$$

Let us treat the left hand side as a polynomial in k. This is

$$k^2 \left(a^2 + 2a + b^2 + 1\right) + k \left((-2n - 3)\left(a^2 + b^2\right) - 2an - 4a - 1\right) + (n + 1)(n + 2)\left(a^2 + b^2\right).$$

Since the coefficient in front of k^2 is positive, we can hope to find n such that this polynomial is positive for all real k. For this the Δ of this quadratic form should be

negative. We have

$$\begin{aligned}\Delta &= \left((-2n-3)\left(a^2+b^2\right) - 2an - 4a - 1\right)^2 \\ &\qquad - 4(n+1)(n+2)\left(a^2+b^2\right)\left(a^2+2a+b^2+1\right) \\ &= -4b^2n^2 + 4(a+2a^2+a^3-2b^2+ab^2)n \\ &\qquad + (1+8a+14a^2+8a^3+a^4-2b^2+8ab^2+2a^2b^2+b^4).\end{aligned}$$

As we can see the leading term is $-4b^2n^2$ and so for big n we get $\Delta < 0$.

We now show that for big n the sequence a_k is strictly log-concave; i.e., $a_k^2 > a_{k+1}a_{k-1}$. This is trivially true for $k = 0$ and $k = n+2$, but it is also easily verified for $k \in \{1, n-1, n, n+1\}$ by just substituting the value of k and letting $n \to \infty$.

To prove the result for $k \in \{2, \ldots, n-2\}$, rewrite the coefficients a_k as:

$$a_k = \binom{n}{k-1}\left[\frac{n-k+1}{k}(a^2+b^2) + \frac{k-1}{n-k+2} - 2a\right].$$

The inequality $a_k^2 > a_{k+1}a_{k-1}$ reduces to

$$\begin{aligned}\binom{n}{k-1}^2\left[\frac{n-k+1}{k}(a^2+b^2) + \frac{k-1}{n-k+2} - 2a\right]^2 \\ > \binom{n}{k}\left[\frac{n-k}{k+1}(a^2+b^2) + \frac{k}{n-k+1} - 2a\right] \\ \times \binom{n}{k-2}\left[\frac{n-k+2}{k-1}(a^2+b^2) + \frac{k-2}{n-k+3} - 2a\right],\end{aligned}$$

This is

$$\begin{aligned}\frac{k}{k-1}\cdot\frac{n-k+2}{n-k+1}\left[\frac{n-k+1}{k}(a^2+b^2) + \frac{k-1}{n-k+2} - 2a\right]^2 \\ > \left[\frac{n-k}{k+1}(a^2+b^2) + \frac{k}{n-k+1} - 2a\right] \\ \times \left[\frac{n-k+2}{k-1}(a^2+b^2) + \frac{k-2}{n-k+3} - 2a\right].\end{aligned}$$

To deal with it we rewrite it slightly.

$$\begin{aligned}
&\left[\frac{n-k+1}{k}(a^2+b^2)+\frac{k-1}{n-k+2}-2a\right]^2 \\
&\quad>\left[\frac{(n-k)}{(k+1)}(a^2+b^2)+\frac{k}{n-k+1}-2a\right] \\
&\qquad\times\left[\frac{n-k+1}{k}(a^2+b^2)+\frac{(k-2)(k-1)(n-k+1)}{(n-k+3)(n-k+2)k}\right. \\
&\qquad\qquad\left.-2a\frac{(k-1)(n-k+1)}{(n-k+2)k}\right].
\end{aligned}$$

For big n and fixed a, b, the right hand side is a product of two positive factors. We shall take the square root of both sides and use the inequality $2\sqrt{xy} \leq x+y$ to bound the right hand side. Then, it is enough to verify:

$$\begin{aligned}
&2\left[\frac{n-k+1}{k}(a^2+b^2)+\frac{k-1}{n-k+2}-2a\right] \\
&\quad>\left[\frac{(n-k)}{(k+1)}(a^2+b^2)+\frac{k}{n-k+1}-2a\right] \\
&\qquad+\left[\frac{n-k+1}{k}(a^2+b^2)+\frac{(k-2)(k-1)(n-k+1)}{(n-k+3)(n-k+2)k}\right. \\
&\qquad\qquad\left.-2a\frac{(k-1)(n-k+1)}{(n-k+2)k}\right].
\end{aligned}$$

Rewrite it by taking the RHS to the LHS and collecting common factors.

$$\begin{aligned}
&\left[\frac{(n-k+1)}{k}-\frac{n-k}{k+1}\right](a^2+b^2)+\frac{2(k-1)}{n-k+2}-\frac{k}{n-k+1} \\
&\quad-\frac{(k-2)(k-1)(n-k+1)}{(n-k+3)(n-k+2)k}-\left[1-\frac{(k-1)(n-k+1)}{(n-k+2)k}\right]2a>0.
\end{aligned}$$

Notice:

- $\dfrac{(n-k+1)}{k}-\dfrac{n-k}{k+1}=\dfrac{n+1}{k(k+1)}$,
- $\dfrac{2(k-1)}{n-k+2}-\dfrac{k}{n-k+1}-\dfrac{(k-2)(k-1)(n-k+1)}{(n-k+3)(n-k+2)k}=$
  $-\dfrac{(1+k+(k-1)^2+n-(k-1)n)(n+1)}{(n-k+1)(n-k+2)(n-k+3)k}$,
- $1-\dfrac{(k-1)(n-k+1)}{(n-k+2)k}=\dfrac{n+1}{(n-k+2)k}$.

Thus, by taking the common denominator, it is enough to verify $P_{a,b}(k,n) > 0$, where

$$\begin{aligned} P_{a,b}(k,n) &= (a^2+b^2)(n-k+3)(n-k+2)(n-k+1) \\ &\quad - \Big(1+k+(k-1)^2+n-(k-1)n\Big)(k+1) \\ &\quad - 2a(k+1)(n-k+3)(n-k+1). \end{aligned}$$

This is a polynomial of degree three in k. The discriminant of a cubic polynomial Ak^3+Bk^2+Ck+D is given by

$$\Delta = B^2C^2 - 4AC^3 - 4B^3D - 27A^2D^2 + 18ABCD,$$

and is negative if there are two conjugate complex and one real roots.

In our case the discriminant of $k \to P_{a,b}(k,n)$ is

$$\Delta(n,a,b) = -4b^2n^6 + O(n^5),$$

and so for big n it is negative (recall that $b \neq 0$). We conclude that there is only one real root. Notice that

$$\begin{aligned} P_{a,b}(2,n) &= (a^2+b^2)n^3 + O(n^2), \\ P_{a,b}(n-2,n) &= n^2 + O(n), \end{aligned}$$

so that for n big enough, $P_{a,b}(k,n) > 0$ for all $k \in [2, n-2]$ as desired.

□

Proof of Proposition 4.22

Proof. Augment the ladder d times to construct a new ladder $\pi' : \Delta^m \to \Delta^w$, where $w < \min\{(k+1)(m+1)^d, \binom{2d+m}{m}\}$. We showed in Proposition 4.11 that π' is a fine and connected ladder and that we can define sets $A_0, \dots, A_k$ as in equation (4.18). We now show that for any state $a \in \Omega'$, it is always possible to move to a different state if $b \in E$ is rolled (except from the state proportional to $p_b^{2d}/C(\boldsymbol{p})$). Fix a state $a \in \Omega'$ in the set A_i, therefore of the form

$$C_a \pi_i(\boldsymbol{p}) \binom{d}{\boldsymbol{n}} \boldsymbol{p}^{\boldsymbol{n}}, \quad \text{for some } \boldsymbol{n} \in \Lambda_d^m.$$

If $\boldsymbol{p}^{\boldsymbol{n}} \neq p_b^d$, then there exists another state $a' \in A_i$ connected to a and such that

$n'_{a',b} = n'_{a,b}+1$ and the chain may move to it. We showed in the proof of Proposition 4.11 that $A_i \cap A_j \neq \emptyset, \forall j \neq i$. Therefore, if $\boldsymbol{p}^{\boldsymbol{n}} = p_b^d$ there exists a connected state a' in $A_j \neq A_i$ such that $n'_{a',b} = n'_{a,b} + 1$, unless $\pi_i(\boldsymbol{p}) \propto p_b^{2d}/C(\boldsymbol{p})$.

Now, consider applying CFTP on the ladder π' using the transition matrix of Proposition 4.15 and the update function of Proposition 4.18. We prove the bound by considering sets of moves that, regardless of the starting point, end up in a singleton. Let a be the minimum of the entries of the matrix V, as produced by Algorithm 17. This choice of a allows us to conclude that whenever we draw $U < a$ in the CFTP algorithm and $B \in E$, then all the tracked particles move, except the particles in the state proportional to $p_b^{2d}/C(\boldsymbol{p})$. Therefore if such event happens on $2d$ consecutive iterations, then the algorithm necessarily ends as all the particles must have coalesced in the state proportional to $p_b^{2d}/C(\boldsymbol{p})$. That is, if $u_1 \leq a, \ldots, u_{2d} \leq a$ we can write

$$\phi_{2d}(i, (b, \ldots, b), (u_1, \ldots, u_{2d})) = \{a\}, \qquad \forall i \in \{0, \ldots, w\},$$

where $a \in \Omega'$ is the state of the ladder proportional to $p_b^{2d}/C(\boldsymbol{p})$. Let τ_b be the number of iterations required for this event to happen for the first time. The probability generating function of τ_b is given by

$$\begin{aligned} f_{\tau_b}(x) &= \sum_{j=0}^{\infty} (ap_b)^{2d} x^{2d} \left[(1-ap_b)x + \ldots + (ap_b)^{2d-1}(1-ap_b)x^{2d}\right]^j \\ &= \frac{x^{2d}(ap_b)^{2d}(ap_b x - 1)}{ap_b x (ap_b x)^{2d} - x(ap_b x)^{2d} + x - 1}, \end{aligned}$$

so that

$$\mathbb{E}[\tau_b] = f'_{\tau_b}(1) = \frac{(ap_b)^{-2d} - 1}{1 - ap_b}.$$

Since the number of required rolls N equals the number of iterations of the algorithm, it follows that $N \leq \tau_b$. The same reasoning holds for all $b \in E$, so that we conclude:

$$\mathbb{E}[N] \leq \min_{b \in E} \mathbb{E}[\tau_b] = \min_{b \in E} \frac{(ap_b)^{-2d} - 1}{1 - ap_b}.$$

□

Proof of Corollary 4.23

Proof. Follows by Proposition 4.22 by noticing that in the case $m = 1$, we necessarily have $E = \{0, 1\}$. □

Proof of Proposition 4.24

Proof. Requiring π to be strictly log-concave is equivalent to have $R_i^2 > R_{i-1}R_{i+1}$ for all $i \in \{1, \dots, k-1\}$ by equation (4.4). In turn, this implies

$$\frac{R_i}{R_{i-1} \vee R_i} \geq \frac{R_{i+1}}{R_i \vee R_{i+1}}, \quad \frac{R_i}{R_i \vee R_{i+1}} \geq \frac{R_{i-1}}{R_{i-1} \vee R_i}, \tag{4.21}$$

so that $\rho \leq 1$ since $P_{i,i+1} \geq P_{i+1,i+2}$ and $P_{i+1,i} \geq P_{i,i-1}$. However, given $p \in (0,1)$, it cannot be that $\rho = 1$. Indeed, this could happen only if $P_{i,i+1} = P_{i+1,i+2}$ and $P_{i+1,i} = P_{i,i-1}$. However, this would imply either $R_i^2 = R_{i-1}R_{i+1}$ or $R_{i+1}^2 = R_i R_{i+2}$ thus contradicting strict log-concavity. We then conclude that $\rho \in (0,1)$ for all $p \in (0,1)$.

Denote by X_t^i the chain at time t given that it started in state i. Monotonic CFTP (cf. Algorithm 16) tracks backwards in time the trajectories of the coupled chains X_t^0 and X_t^k and stops when the two coalesce. Following the notation of [52], let $T_\star$ be the time this happens and, to ease the analysis, define $T^\star$ as the smallest time such that $X_t^0 = X_t^k$, where the chains are now tracked forwards in time. Notice that $T_\star$ and $T^\star$ have the same distribution and that the number of tosses N required by the algorithm equals $T_\star$.

Define $D_t^{i,j} = |X_t^i - X_t^j|$ as the distance between two coupled particles started at states i and j after t steps. In particular, focus on the distance $D_t^{i,i+1}$ between two particles started at consecutive states. At each step a p-coin is tossed and a uniform random variable is drawn so that the trajectories of the two chains can be tracked in a coupled fashion. In particular, given equation (4.21), we have that the two particles started at states i and $(i+1)$ can in one step either stay still, coalesce in state i or state $(i+1)$, move to states $(i+1)$ and $(i+2)$ or move to states $(i-1)$ and i respectively. Therefore, after one step the distance between the two coupled and consecutive particles can either decrease by 1 or remain the same:

$$D_1^{i,i+1} = \begin{cases} 0 & \text{with probability } (P_{i,i+1} - P_{i+1,i+2}) + (P_{i+1,i} - P_{i,i-1}) \\ 1 & \text{with probability } 1 - (P_{i,i+1} - P_{i+1,i+2}) - (P_{i+1,i} - P_{i,i-1}) \end{cases}$$

where the transition probabilities $P_{i,j}$ are given in equation (4.11). Denote by

$$\rho_{i,i+1} = 1 - (P_{i,i+1} - P_{i+1,i+2}) - (P_{i+1,i} - P_{i,i-1}),$$

so that $\mathbb{E}[D_1^{i,i+1}] = \rho_{i,i+1}$. Let $\rho = \max_i \rho_{i,i+1}$ and notice that by conditioning on

how the particles move on the first step and by the Markov property, it follows

$$\begin{aligned}\mathbb{E}[D_t^{i,i+1}] &= P_{i+1,i+2}\mathbb{E}[D_{t-1}^{i+1,i+2}] + P_{i,i-1}\mathbb{E}[D_{t-1}^{i-1,i}] + \left(1 - P_{i,i+1} - P_{i+1,i}\right)\mathbb{E}[D_{t-1}^{i,i+1}] \\ &\leq (\mathbb{E}[D_{t-1}^{i-1,i}] \vee \mathbb{E}[D_{t-1}^{i,i+1}] \vee \mathbb{E}[D_{t-1}^{i+1,i+2}])\rho_{i,i+1} \\ &\leq \rho^t,\end{aligned}$$

where $\vee$ denotes the maximum between two numbers.

To conclude, notice that $\mathbb{P}(T^\star \geq t) = \mathbb{P}(D_t^{0,k} \geq 1)$. It then follows by Markov's inequality and the result above that

$$\mathbb{P}(T^\star \geq t) = \mathbb{P}(D_t^{0,k} \geq 1) \leq \mathbb{E}[D_t^{0,k}] = \sum_{i=0}^{k-2} \mathbb{E}[D_t^{i,i+1}] \leq (k-1)\rho^t,$$

as desired. □

Proof of Lemma 4.26

Proof. Note that a univariate ladder is log-concave if its coefficients R_i define a log-concave sequence. Then, let R be a random variable on $\{0, \ldots, k\}$ having p.m.f. proportional to the coefficients R_i of the ladder π, that is such that $\mathbb{P}(R = i) \propto R_i$. Moreover, let n be such that $Z = R + B_n$ is strictly log-concave, as stated in Theorem 4.21. Consider $\pi' : (0,1) \to \Delta^{k+n}$, an n-fold augmentation of π. As noticed in Remark 4.10, $Y \sim \pi'(p)$ has the same distribution as $\pi + \mathrm{Bin}(n,p)$ and the coefficients R_i's of the ladder π' are proportional to $\mathbb{P}(Z = i)$. The desired result holds by noticing that multiplication by a constant preserves log-concavity. □

Glossary

a.s. almost surely.

CFTP Coupling From The Past.

eq. equation.

i.i.d. independent and identically distributed.

LHS left hand side.

MCMC Markov Chain Monte Carlo.

pmf probability mass function.

r.v. random variable.

RHS right hand side.

w.l.o.g. without loss of generality.

Bibliography

[1] Asmussen, S., Glynn, P. W., and Thorisson, H. (1992). Stationarity detection in the initial transient problem. *ACM Transactions on Modeling and Computer Simulation 2*(2), 130–157.

[2] Bhatia, R. and Davis, C. (2000). A better bound on the variance. *The American Mathematical Monthly 107*(4), 353–357.

[3] Blanchet, J. and Meng, X. (2005). Exact sampling, regeneration and minorization conditions. Technical report, Tech. rep., Columbia University. `http://web.stanford.edu/~jblanche/papers/JSMsent.pdf`.

[4] Blanchet, J. and Zhang, F. (2020, Dec). Exact simulation for multivariate itô diffusions. *Advances in Applied Probability 52*(4), 1003–1034.

[5] Bubley, R. and Dyer, M. (1997, Oct). Path coupling: A technique for proving rapid mixing in markov chains. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, 223–231.

[6] Cai, Y., Oikonomou, A., Velegkas, G., and Zhao, M. (2020). An efficient ε-bic to bic transformation and its application to black-box reduction in revenue maximization. `arXiv:1911.10172`.

[7] Dagum, P., Karp, R., Luby, M., and Ross, S. (2000). An optimal algorithm for Monte Carlo estimation. *SIAM Journal on Computing 29*(5), 1484–1496.

[8] Dale, H., Jennings, D., and Rudolph, T. (2015). Provable quantum advantage in randomness processing. *Nature communications 6*, 8203.

[9] Dempster, A. P. (1966). New Methods for Reasoning Towards Posterior Distributions Based on Sample Data. *The Annals of Mathematical Statistics 37*(2), 355 – 374.

[10] Dempster, A. P. (1972). A class of random convex polytopes. *The Annals of Mathematical Statistics 43*(1), 260–272.

[11] DeRose, T. D. (1988, July). Composing bézier simplexes. *ACM Transactions on Graphics 7*(3), 198–221.

[12] Dughmi, S., Hartline, J. D., Kleinberg, R., and Niazadeh, R. (2017). Bernoulli factories and black-box reductions in mechanism design. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 158–169.

[13] Fill, J. A. et al. (1998). An interruptible algorithm for perfect sampling via markov chains. *The Annals of Applied Probability 8*(1), 131–162.

[14] Flajolet, P., Pelletier, M., and Soria, M. (2011). On buffon machines and numbers. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete algorithms*, 172–183. Society for Industrial and Applied Mathematics.

[15] Flegal, J. M. and Herbei, R. (2012). Exact sampling for intractable probability distributions via a Bernoulli factory. *Electronic Journal of Statistics 6*, 10–37.

[16] Gonçalves, F. B., Łatuszyński, K., Roberts, G. O., et al. (2017). Barker's algorithm for bayesian inference with intractable likelihoods. *Brazilian Journal of Probability and Statistics 31*(4), 732–745.

[17] Gonçalves, F. B., Łatuszyński, K. G., and Roberts, G. O. (2020). Exact monte carlo likelihood-based inference for jump-diffusion processes. `arXiv:1707.00332`.

[18] Goyal, V. and Sigman, K. (2012). On simulating a class of Bernstein polynomials. *ACM Transactions on Modeling and Computer Simulation 22*(2), Art. 12, 5.

[19] Henderson, S. G. and Glynn, P. W. (2003). Nonexistence of a class of variate generation schemes. *Operations Research Letters 31*(2), 83–89.

[20] Herbei, R. and Berliner, L. M. (2014). Estimating ocean circulation: an MCMC approach with approximated likelihoods via the Bernoulli factory. *Journal of the American Statistical Association 109*(507), 944–954.

[21] Hoggar, S. G. (1974). Chromatic polynomials and logarithmic concavity. *Journal of Combinatorial Theory. Series B 16*, 248–254.

[22] Holtz, O., Nazarov, F., and Peres, Y. (2011). New Coins from Old, Smoothly. *Constructive Approximation 33*(3), 331–363.

[23] Huber, M. (2016a). Nearly optimal bernoulli factories for linear functions. *Combinatorics, Probability and Computing 25*(4), 577–591.

[24] Huber, M. (2017). Optimal Linear Bernoulli Factories for Small Mean Problems. *Methodology and Computing in Applied Probability 19*(2), 631–645.

[25] Huber, M. (2019). Designing perfect simulation algorithms using local correctness. arXiv:1907.06748.

[26] Huber, M. L. (2016b). *Perfect simulation*, Volume 148 of *Monographs on Statistics and Applied Probability*. CRC Press, Boca Raton, FL.

[27] Jacob, P. E. and Thiery, A. H. (2015). On nonnegative unbiased estimators. *Annals of Statistics 43*(2), 769–784.

[28] Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. arXiv:1611.01144.

[29] Johnson, O. and Goldschmidt, C. (2006). Preservation of log-concavity on summation. *ESAIM. Probability and Statistics 10*, 206–215.

[30] Keane, M. and O'Brien, G. L. (1994). A bernoulli factory. *ACM Transactions on Modeling and Computer Simulation (TOMACS) 4*(2), 213–219.

[31] Krantz, S. G. and Parks, H. R. (2002). *A primer of real analytic functions* (Second ed.). Birkhäuser Advanced Texts: Basler Lehrbücher. [Birkhäuser Advanced Texts: Basel Textbooks]. Birkhäuser Boston, Inc., Boston, MA.

[32] Łatuszyński, K., Kosmidis, I., Papaspiliopoulos, O., and Roberts, G. O. (2011). Simulating events of unknown probabilities via reverse time martingales. *Random Structures Algorithms 38*(4), 441–452.

[33] Lee, A., Doucet, A., and Łatuszyński, K. (2014). Perfect simulation using atomic regeneration with application to sequential monte carlo. arXiv:1407.5770.

[34] Lugosi, G. and Mendelson, S. (2019). Mean estimation and regression under heavy-tailed distributions–a survey. arXiv:1906.04280.

[35] Lyne, A.-M., Girolami, M., Atchadé, Y., Strathmann, H., Simpson, D., et al. (2015). On russian roulette estimates for bayesian inference with doubly-intractable likelihoods. *Statistical science 30*(4), 443–467.

[36] Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. arXiv:1611.00712.

[37] McLeish, D. (2011). A general method for debiasing a Monte Carlo estimator. *Monte Carlo Methods and Applications 17*(4), 301–315.

[38] Mendo, L. (2019). An asymptotically optimal bernoulli factory for certain functions that can be expressed as power series. *Stochastic Processes and their Applications 129*(11), 4366–4384.

[39] Mendo, L. (2020). Simulating a coin with irrational bias using rational arithmetic. arXiv:2010.14901.

[40] Mira, A. (2001). Ordering and improving the performance of Monte Carlo Markov chains. *Statistical Science. A Review Journal of the Institute of Mathematical Statistics 16*(4), 340–350.

[41] Morina, G., Latuszynski, K., Nayar, P., and Wendland, A. (2019). From the bernoulli factory to a dice enterprise via perfect sampling of markov chains. arXiv:1912.09229.

[42] Mossel, E., Peres, Y., and Hillar, C. (2005). New coins from old: Computing with unknown bias. *Combinatorica 25*(6), 707–724.

[43] Nacu, Ş. and Peres, Y. (2005, 02). Fast simulation of new coins from old. *Annals of Applied Probability 15*(1A), 93–115.

[44] Niazadeh, R. (2017). *PhD Thesis: Algorithms Vs. Mechanisms.: Mechanism Design for Complex Environments*. Ph. D. thesis, Cornell University.

[45] Niazadeh, R., Leme, R. P., and Schneider, J. (2020). Combinatorial bernoulli factories: Matchings, flows and other polytopes. arXiv:2011.03865.

[46] Papaspiliopoulos, O. (2011). *Monte Carlo probabilistic inference for diffusion processes: a methodological framework*, 82–103. Cambridge University Press.

[47] Patel, R. B., Rudolph, T., and Pryde, G. J. (2019). An experimental quantum bernoulli factory. *Science advances 5*(1), eaau6668.

[48] Peres, Y. (1992). Iterating von neumann's procedure for extracting random bits. *The Annals of Statistics 20*(1), 590–597.

[49] Peskun, P. H. (1973). Optimum Monte-Carlo sampling using Markov chains. *Biometrika 60*, 607–612.

[50] Pólya, G. (1928). Über positive darstellung von polynomen. *Vierteljahresschrift der Naturforschenden Gesellschaft in Zurich 73*, 141–145.

[51] Powers, V. and Reznick, B. (2001). A new bound for pólya's theorem with applications to polynomials positive on polyhedra. *Journal of Pure and Applied Algebra 164*(1), 221 – 229. Effective Methods in Algebraic Geometry.

[52] Propp, J. G. and Wilson, D. B. (1996). Exact sampling with coupled markov chains and applications to statistical mechanics. *Random Structures & Algorithms 9*(1-2), 223–252.

[53] Rhee, C.-H. and Glynn, P. W. (2015). Unbiased Estimation with Square Root Convergence for SDE Models. *Operations Research 63*(5), 1026–1043.

[54] Saumard, A. and Wellner, J. A. (2014). Log-concavity and strong log-concavity: a review. *Statistics Surveys 8*, 45–114.

[55] Schmon, S. M., Doucet, A., and Deligiannidis, G. (2019). Bernoulli race particle filters. `arXiv:1903.00939`.

[56] Sinha, D., Sankararama, K. A., Kazerouni, A., and Avadhanula, V. (2020). Multi-armed bandits with cost subsidy. `arXiv:2011.01488`.

[57] Sison, C. P. and Glaz, J. (1995). Simultaneous confidence intervals and sample size determination for multinomial proportions. *Journal of the American Statistical Association 90*(429), 366–369.

[58] Thomas, A. C. and Blanchet, J. H. (2012). A practical implementation of the bernoulli factory. `arXiv:1106.2508`.

[59] Vats, D., Gonçalves, F., Łatuszyński, K., and Roberts, G. O. (2020). Efficient bernoulli factory mcmc for intractable posteriors. `arXiv:2004.07471`.

[60] Von Neumann, J. (1951). Various techniques used in connection with random digits. In A. Householder, G. Forsythe, and H. Germond (Eds.), *Monte Carlo Method*, 36–38. Washington, D.C.: U.S. Government Printing Office: National Bureau of Standards Applied Mathematics Series, 12.

[61] Wästlund, J. (1999). Functions arising by coin flipping. `http://www.math.chalmers.se/~wastlund/coinFlip.pdf`.

[62] Yuan, X., Liu, K., Xu, Y., Wang, W., Ma, Y., Zhang, F., Yan, Z., Vijay, R., Sun, L., and Ma, X. (2016). Experimental quantum randomness processing using superconducting qubits. *Physical review letters 117*(1), 010502.